AUDIO-VIDEO BASED HANDWRITTEN MATHEMATICAL CONTENT RECOGNITION

A Thesis
Presented to
The Academic Faculty

by

Smita Vemulapalli

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2012

AUDIO-VIDEO BASED HANDWRITTEN MATHEMATICAL CONTENT RECOGNITION

Approved by:

Professor Monson H. Hayes III,
Advisor
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Professor John R. Barry
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Professor Chin-Hui Lee
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Professor Karsten Schwan
College of Computing
Georgia Institute of Technology

Professor David V. Anderson
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Date Approved: 31 October 2012

To

My father, Mr. Ramu Vemulapalli.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my Ph.D. advisor Prof. Monson Hayes for his encouragement, support and guidance during the course of my research work. While he always encouraged me to become an independent thinker, the questions that he posed to me during our meetings made me think harder and helped me crystallize the ideas into a structured research work.

I would also like to thank Prof. Chin-Hui Lee, Prof. David V. Anderson and Prof. John R. Barry for their valuable feedback during the early stages of this research work. I would also like to thank Prof. Karsten Schwan for his willingness to be on the Ph.D. defense committee and for bringing in a very distinct point of view on the work presented in this dissertation.

I would also like to acknowledge the support of Texas Instruments, who, via the Texas Instruments Leadership University (TILU) program, were the primary sponsors of my research at Georgia Tech. The regular feedback that I received from the team at Texas Instruments was instrumental in driving some key components of this dissertation. My friends at Georgia Tech and my colleagues at the Center for Signal & Image Processing (CSIP) deserve a special mention. I feel honored to have been a part of the Georgia Tech community and the CSIP research group.

Finally, I would like to thank my parents Ramu Vemulapalli and Sita Devi Vemulapalli and sister Sandhya Vemulapalli for the encouragement and love that has helped me through the years that I have spent as a graduate student. Last but not the least, my husband, Vibhore deserves a special mention for the love, support and advice.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Recognizing handwritten mathematical content is a challenging problem, and more so when such content appears in classroom videos. However, given the fact that in such videos the handwritten text and the accompanying audio refer to the same content, a combination of video and audio based recognizers has the potential to significantly improve the recognition accuracy for handwritten mathematical content that appears in such videos. This dissertation, using a combination of video and audio based recognizers, focuses on improving the accuracy associated with the recognition of handwritten mathematical content in classroom videos.

Our approach makes use of a video recognizer as the primary recognizer and a multi-stage assembly, developed as part of this research, is used to facilitate effective combination with an audio recognizer. Specifically, the multi-stage assembly addresses the following challenges related to audio-video based handwritten mathematical content recognition: (1) Video Preprocessing - generates a timestamped sequence of segmented characters from the classroom video in the face of occlusions and shadows caused by the instructor, (2) Ambiguity Detection - determines the subset of input characters that may have been incorrectly recognized by the video based recognizer and forwards this subset for disambiguation using the audio based recognizer, (3) A/V Synchronization - establishes correspondence between the handwritten character and the spoken content, the occurrences of which may be missing and/or appear at significantly different time instances in the input, (4) A/V Combination - combines the synchronized outputs from the video and audio based recognizers and generates the final recognized character, and (5) Grammar Assisted A/V Based Mathematical Content Recognition - utilizes a base mathematical speech grammar for

both character and structure disambiguation. Experiments conducted using videos recorded in a classroom-like environment demonstrate the significant improvements in recognition accuracy that can be achieved using our techniques.

CHAPTER I

INTRODUCTION

Recent years have witnessed a rapid increase in the number of e-learning and advanced learning initiatives that either use classroom videos as the primary medium of instruction or make them available online for reference by the students. As the volume of such recorded video content increases, it is amply clear that in order to efficiently navigate through the available classroom videos there is a need for techniques that can help extract, identify and summarize the content in such videos. In this context, and given the fact that the whiteboard continues to be the preferred and effective medium for teaching complex mathematical and scientific concepts, this dissertation focuses on improving the accuracy associated with the recognition of handwritten mathematical content in classroom videos.

1.1 Background & Challenges

There is a significant body of research devoted to the topic of handwritten mathematical content recognition [3, 14, 126, 107, 69, 2], and to the extraction and recognition of textual content from videos [100, 92, 63, 36, 37]. While the research conducted as part of this dissertation is closely related and dependent on the advances made in the aforementioned fields, its focus on the use of an audio based recognizer in combination with a primary, video based recognizer to improve the recognition of handwritten mathematical content from classroom videos presents a range of new and interesting challenges that span across the various components of the recognition system. This dissertation focuses on the challenges associated with the use of a combination of audio and video based recognizers for improving the handwritten mathematical content recognition accuracy in such videos. Specifically, we address the following challenges:

- *How to extract timestamped handwritten characters from classroom videos?* - The task of extracting timestamped handwritten characters from classroom videos presents significant challenges in the form of occlusions and shadows caused by the instructor, non-uniform illumination and erasures of the white-board. While the accuracy of the segmentation is critical to the accuracy of the video based handwritten character recognizer, the accuracy of determining the timestamp associated with the first appearance of a handwritten character is equally critical in establishing the correspondence between the handwritten and the spoken content, which is required for enabling audio-video based handwritten mathematical content recognition.

- *How to establish correspondence between the handwritten and the spoken content?* - The first step towards combining the output of video and audio based recognizers is to establish a correspondence between the handwritten and the spoken content. The need for establishing correspondence is not a commonly encountered situation in classifier combination literature, where the classifiers generate outputs which correspond to the same underlying entity (*e.g.* combining output from two character recognizers operating on the same set of segmented characters). Moreover, in the face of occlusions, shadows and the fact that the output from the individual recognizers, operating on the noisy input classroom video, may not be very accurate, establishing a correspondence between spoken and handwritten content can be a very challenging problem.

- *How to combine the outputs of the audio and the video recognizers?* - The challenges associated with combining the output from the audio and the video recognizers can be attributed to a couple of reasons. First, the recognition accuracy of one recognizer may be better than the other, or better than the other for certain specific characters or specific instructors. Second, the match

scores generated by the recognizers may not be normalized with respect to each other. The combination techniques should be robust to such variations in the recognition accuracies and the match scores.

- *How to limit the errors introduced by combination?* - In the proposed solution, the video based recognizer acts as the primary recognizer and the audio based recognizer is used to corroborate or contradict the output generated by the video based recognizer. It is often times possible that a combination with audio based recognizer may introduce more errors than corrections in the video recognizer's output and, as a result, deteriorate the recognition accuracy as compared to the accuracy of the standalone video based recognizer. This motivates the need to determine a subset of characters, from the output of the video based recognizer, which are more likely to be erroneous and only forward these characters for subsequent combination.

- *How can mathematical speech grammar be used to assist in character & structure disambiguation?* - Traditionally, for handwritten mathematical content recognition, the recognizer relies on character segmentation, a character recognizer and a structure recognizer. In classroom videos, the presence of accompanying audio content spoken by the instructor and the mathematical grammar associated with spoken content offers another interesting means for disambiguating the character recognition results and the structure associated with the mathematical content (*e.g.* differentiating between the exponent `'square'` and the numeral `'two'`).

1.2 Thesis Statement

"The recognition accuracy for handwritten mathematical content in classroom videos can be significantly improved by using a combination of audio and video based recognizers."

1.3 Solution Approach

Our approach makes use of a video text recognizer, which acts as the primary recognizer and is responsible for character segmentation and the generation of *location coordinates*, a *video timestamp* and a set of *video options* for every segmented character from the input classroom video. Here, the *video timestamp* corresponds to the first visible occurrence of the segmented character in the input video, the *location coordinates* represent the spatial location of the segmented character in the video frame and the *video options* correspond to the set of likely character recognition outputs for the segmented character in question. Each video option, in addition to containing the recognized *character name*, also contains a *video match score*, which represents the recognizer's belief in the correctness of the character name as the output for the corresponding segmented character.

In some cases the match score for the top video option may be high enough, in absolute or relative terms, to accept the corresponding character name as the recognized output. However, in other cases the choice for the final recognized output may not be clear and we consider such characters to be *ambiguous*. For characters that are determined to be ambiguous, our approach makes use of an audio text recognizer to locate, for each video option of an ambiguous character, a set of candidate *audio options*. Each audio option, here, corresponds to an occurrence of the video option's character name in the audio component of the classroom video and consists of an *audio search term*, an *audio timestamp* and an *audio match score*. The audio search term corresponds to a specific spoken representation (*e.g.* 'square') of the video option's character name (*e.g.* '2'), the audio timestamp indicates the time at which the audio occurrence was found in the video and the audio match score represents the audio recognizer's belief in the occurrence of the audio search term in the video at the location indicated by the audio timestamp. For each video option, the set of candidate audio options is subsequently pruned using synchronization techniques to limit the

set to at most one best *synchronized* audio option. The resulting set of synchronized video and audio options, for each segmented character, is then submitted to the combination component. The combination component, based on the video match scores, the audio match scores and other factors like the accuracy of the video and the audio recognizers generates the final recognized characters, one for each segmented character. Note that the combination component can be configured to generate a ranked list of recognition results for each segmented character as well. Such options can be utilized by the grammar assisted audio-video based content disambiguation stage, which is described next.

The grammar assisted audio-video based content disambiguation stage makes use of speech grammar to assist in further disambiguation of the character recognition results generated by the audio-video based combination component and, by means of audio cues derived from speech recognition results, aids in structure analysis as well. It may be noted that this stage can be easily configured to disambiguate between the character recognition results that are directly obtained from the video text recognizer. The remainder of this section briefly summarizes the main contributions of this dissertation:

- *Video Preprocessing* - We make use of a combination of traditional video and image processing techniques to address the issues caused by occlusions, shadows, non-uniform illumination and erasures, and extract and segment handwritten characters from classroom videos. We also propose a technique to determine, for each segmented character, the timestamp associated with its first appearance in the video.

- *Ambiguity Detection & Option Selection* - We evaluate several different techniques, including the use of preset absolute and relative thresholds that operate

on the video match scores and techniques based on character-specific thresholds computed from the training data set, to detect the characters that have a higher chance of being incorrectly recognized by the video text recognizer. An optimal subset of video options, for each such character, is then forwarded for audio-video synchronization and combination.

- *Audio-Video Synchronization* - We make use of a combination of multiple features like the audio match scores, the time difference between the video timestamp and the audio timestamp, the number of matching pairs of handwritten characters and audio utterances written and spoken in the vicinity of the character that is being synchronized, etc. to establish a correspondence between the handwritten and the spoken content.

- *Audio-Video Combination* - This dissertation proposes, implements and evaluates several techniques for combining the synchronized output of the audio and the video based recognizers. Besides using traditional measurement and rank level combination methods, we have extended such techniques to incorporate recognizer-specific weights, character-specific weights for each recognizer and have also used an ensemble of recognizers for achieving better character recognition accuracy.

- *Grammar Assisted Audio-Video Based Content Disambiguation* - Our approach uses the information from the video and the audio content and a base mathematical speech grammar to determine an intermediate constrained grammar that is specific to the content being recognized. This intermediate grammar is then used by a speech recognizer to recognize the mathematical content and the associated structure.

1.4 Organization of this Dissertation

The remainder of this dissertation is organized as follows:

Chapter 2 provides an overview of the various components of the end-to-end recognition solution, and describes the mathematical model and the data set.

Chapter 3 describes the video and image processing techniques used to extract segmented and timestamped handwritten characters from classroom videos.

Chapter 4 describes a number of techniques for detecting ambiguous characters and for selecting the video options to be used for combination.

Chapter 5 describes various audio-video synchronization techniques that are based on features derived from the recognized video and the audio content.

Chapter 6 describes several audio-video combination techniques for combining the synchronized video and the audio options.

Chapter 7 describes a grammar assisted audio-video based mathematical content recognition technique and its application to character and structure disambiguation.

Chapter 8 presents a summary of the research across several related specializations, including text recognition, speech recognition and classifier combination.

Chapter 9 concludes by summarizing the key contributions and findings of this dissertation, and provides an outline for future work.

CHAPTER II

SYSTEM OVERVIEW

This chapter provides an overview of the end-to-end system that enables audio-video based handwritten mathematical content recognition from classroom videos and sets up the context for describing the recognition components and techniques presented in the subsequent chapters. This chapter, besides presenting the overview, also introduces a preliminary mathematical model that encapsulates the end-to-end recognition system, describes the character recognizer and the audio recognizer used by the end-to-end recognition system [115, 114], and provides the details of the data set used for training and evaluation.

2.1 End-to-End Recognition System

The end-to-end system for handwritten mathematical content recognition from classroom videos, shown in Figure 1, is organized as a multi-stage assembly. It consists of three distinct stages: the video preprocessing stage, the audio-video based character recognition stage and the grammar assisted audio-video based content disambiguation stage.

The *video preprocessing stage* includes all processing required to extract regions

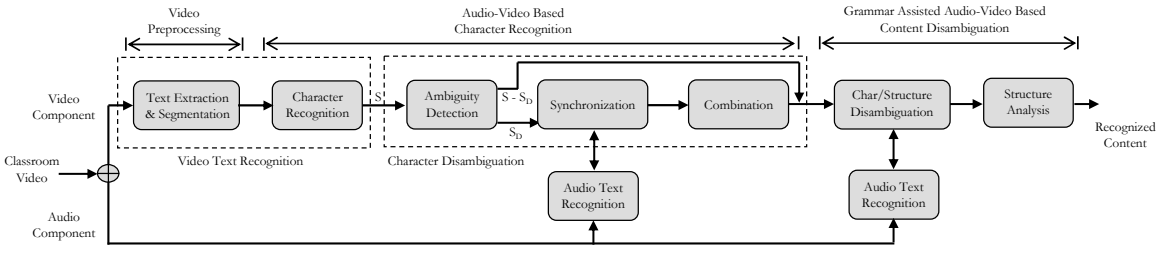


Figure 1: System overview

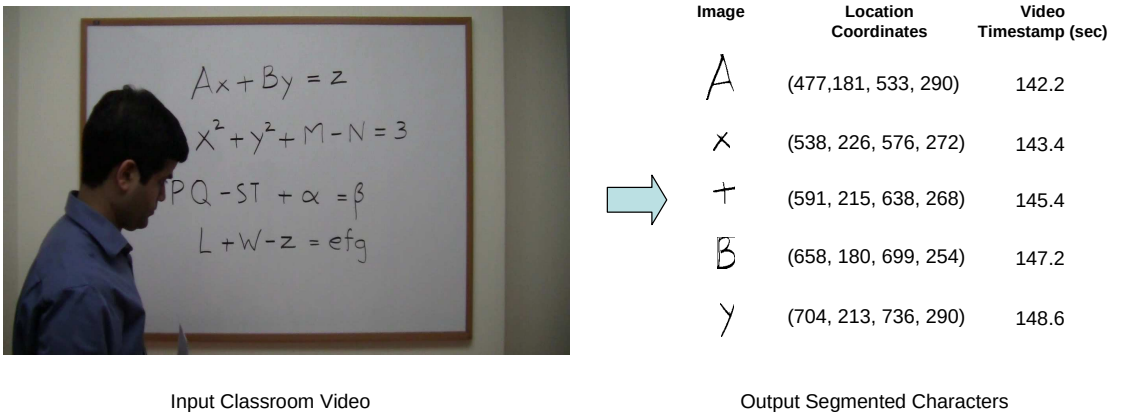


Figure 2: Input and output of the video preprocessing stage

of text containing handwritten mathematical content from the video, segment characters and also generate information such as the timestamp corresponding to the first appearance of the handwritten character in the video and the location of each segmented character in the video frame. The input and the output for the video preprocessing stage are shown in Figure 2. More details about the various components that constitute the video preprocessing stage will be presented in Chapter 3.

The *audio-video based character recognition stage* is responsible for recognizing the segmented characters generated by the video preprocessing stage using a character recognizer and performing audio-assisted character disambiguation on a selected subset of such recognized characters. The video preprocessing stage and the character recognition component from the audio-video based character recognition stage together constitute the *video text recognizer*. For each segmented character, the *video text recognizer* output consists of the video timestamp, the location of the segmented character and a set of *video options*, where each video option consists of a possible *character name* and a *video match score* that represents the recognizer's belief in the correctness of the *character name* as the output for the corresponding segmented character. Following the video text recognition step, the task of audio-assisted character disambiguation, in our system, is performed by three distinct components:

- *the ambiguity detection component* - attempts to determine, for each segmented character, the correctness of the video options generated by the video text recognizer and forwards only a subset of characters that it determines to have a high chance of being erroneous, termed *ambiguous characters*, to the subsequent character disambiguation components.

- *the audio-video synchronization component* - for an ambiguous segmented character, our system makes use of an *audio text recognizer* to locate, for each video option of the segmented character, a set of candidate *audio options*. Each audio option, here, represents an occurrence of the *audio search term* that corresponds to the video option's character name, in the audio component of the video. The audio option consists of the *audio search term*, an *audio timestamp* that indicates the time at which the audio search term was found in the classroom video and an *audio match score* that represents audio recognizer's belief in the occurrence of the audio search term in the video at the location indicated by the audio timestamp. The audio-video synchronization component, for each video option, is responsible for pruning the set of audio options such that there is at most one audio option remaining for each video option.

- *the audio-video combination component* - combines the output of the video text recognizer and the synchronized output from the audio text recognizer to determine the final recognized character for each segmented character that was earlier determined to be ambiguous by the ambiguity detection component.

The *grammar assisted audio-video based content disambiguation stage* uses the character recognition results from the audio-video based combination stage (or directly from the video text recognizer) and initial speech recognition results from an audio text recognizer, configured with a base mathematical speech grammar, to determine a constrained speech grammar that is more specific to the content being

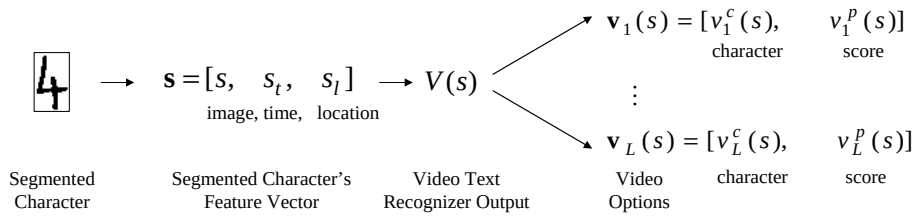


Figure 3: Segmented character, the corresponding feature vector, the video text recognizer output and the corresponding set of video options

recognized. The audio text recognizer, configured with the constrained speech grammar, is then used to arrive at the final character recognition results and audio cues from the speech recognition results are also used to assist in structure analysis.

2.2 Preliminary Mathematical Model

In this section we describe the preliminary mathematical model that is used to represent the end-to-end recognition system. In the following chapters, we will be utilizing and progressively augmenting this mathematical model to assist us in the description of the various components of the recognition system.

Dictionary. The dictionary of symbols that are to be recognized will be denoted by C, and a specific symbol within the dictionary will be denoted by c. It is assumed that the dictionary contains L characters.

Character Segmentation & Ground Truth. Each character that is extracted from the video by the text extraction and segmentation module will be denoted by s, which is the actual image of the character as represented on the left of Figure 3. Associated with each character s, is a video timestamp, s_t, that indicates the time at which the character is first detected in the video, and the location, s_l, of the character s in the video frame. The character s along with s_t and s_l, constitute a feature vector for the segmented character,

$$\mathbf{s} = [s, s_t, s_l] \tag{1}$$

For the purpose of evaluation, we define S to be the set of segmented characters s from a given test data set of videos and we define a function G that returns the ground truth $G(s)$ for a given character $s \in S$, i.e.,

$$G(s) = c \tag{2}$$

where c is the actual character corresponding to the segmented character s.

Finally, we also define an Equals function $E(c_1, c_2)$ given by

$$E(c_1, c_2) = \begin{cases} 1, & \text{if } c_1 = c_2 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where $c_1, c_2 \in C$.

Character Recognition. When a character is passed to the character recognition component, a set of video option vectors are produced, $\mathbf{v}_j(s), j = 1, 2, \ldots, L$, and we represent this process by the function V,

$$V(s) = [\mathbf{v}_1(s), \mathbf{v}_2(s), \ldots, \mathbf{v}_L] \tag{4}$$

Each of these video options is an ordered pair

$$\mathbf{v}_j(s) = [v_j^c(s), v_j^p(s)] \tag{5}$$

where $v_j^c(s)$ is a character from the dictionary C, and $v_j^p(s)$ is the video match score that is generated by the character recognizer. The video options are assumed to be arranged in decreasing order of their video match scores. Figure 3 shows a segmented character s, the feature vector, and the set of video options that are generated by the video text recognizer.

For a given segmented character s, its timestamp s_t, the location coordinates s_l and the set of video options $V(s)$ are forwarded to the subsequent stages as the output

of the video text recognizer. Without any audio cues, the first video option $\mathbf{v}_1(s)$ is used as the recognition output. The character recognition accuracy $\alpha_V(S)$ of the video text recognizer for the complete test data set S can therefore be calculated as

$$\alpha_V(S) = \frac{\text{\# of Correct First Video Options}}{\text{Total Number of Characters}} = \frac{\sum_{\forall s \in S} E(v_1^c(s), G(s))}{|S|} \tag{6}$$

Ambiguity Detection. As mentioned in the previous section, not all characters recognized by the video text recognizer are forwarded for a subsequent combination with the output of the audio text recognizer. The decision to forward the video text recognizer results for combination is done by the ambiguity detection component, which in our model is represented as a function D operating on the set of segmented characters S. The function $D(S)$ produces S_D, which is a subset of characters from the set S that are categorized as ambiguous based on various conditions imposed on the output of the video text recognizer V. The set of non-ambiguous characters is, therefore, $S - S_D$. Chapter 4 describes a set of ambiguity detection techniques and augments the mathematical model with a set of option selection techniques, which for a given ambiguous character determine the subset of video options that should be forwarded for audio-video combination.

Audio Text Recognition. In preparation for performing audio-video based synchronization and combination to recognize the ambiguous segmented characters $s \in S_D$, the audio text recognizer is invoked by the A/V synchronization component (described next), once for each video option $\mathbf{v}_j(s)$ generated by the video text recognizer $V(s)$. The output of audio text recognizer for the jth video option corresponding to a segmented character s is a set of M audio option vectors, $\mathbf{a}_{j,k}(s), k = 1, 2, \ldots, M$ and we use $A_j(s)$ to denote this set,

$$A_j(s) = [\mathbf{a}_{j,1}(s), \mathbf{a}_{j,2}(s), \ldots, \mathbf{a}_{j,k}(s)] \tag{7}$$

Each of the audio options contained in the set $A_j(s)$ is an ordered set

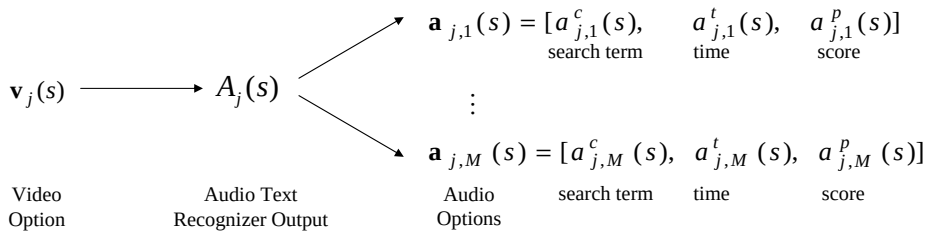


Figure 4: A single video option, the audio text recognizer output and the corresponding set of audio options

$$\mathbf{a}_{j,k}(s) = [a^c_{j,k}(s), a^t_{j,k}(s), a^p_{j,k}(s)] \tag{8}$$

Here, $\mathbf{a}_{j,k}(s)$, denotes the kth audio option The first element in this vector, $a^c_{j,k}(s)$, is the audio search term that is the audio equivalent of $v^c_j(s)$, the second, $a^t_{j,k}(s)$, is the audio timestamp, and the third, $a^p_{j,k}(s)$, is the audio match score. The relationship between a video option and the corresponding set of audio options is shown in Figure 4.

A/V Synchronization. The A/V synchronization component receives the output from the video text recognizer. For each video option $\mathbf{v}_j(s)$, the synchronization component invokes the audio text recognizer, which in response generates a set of audio options $A_j(s)$. Now, the task of the synchronization component, represented as a function Y operating on the set of audio options $A_j(s)$, is to select an audio option $\mathbf{a}_{j,k'}(s) \in A_j(s)$ as follows.

$$Y(A_j(s)) = \mathbf{a}_{j,k'}(s) \tag{9}$$

Here $\mathbf{a}_{j,k'}(s)$ is the audio option that is considered to be best synchronized with the video options $\mathbf{v}_j(s)$. k' is the index of the audio option in the set $A_j(s)$.

A/V Combination. Similar to other components of the A/V based character disambiguation stage, the A/V combination component is also represented as a function. The A/V combination function Z operates on segmented character $s \in S_D$ and

is defined as

$$Z(s) = \mathbf{v}_{j'}(s) \tag{10}$$

where, $\mathbf{v}_{j'}(s) \in V(s)$ and j' is the index of the video option that is considered to be correct by the system at the end of the A/V combination process. The character recognition accuracy $\alpha_Z(S_D)$, calculated at the end of the A/V combination step over the set of ambiguous segmented characters S_D can be calculated as follows.

$$\alpha_Z(S_D) = \frac{\#\text{Ambiguous Characters with } Z(s) \text{ Correct}}{\text{Total Number of Ambiguous Characters}} = \frac{\sum_{\forall s \in S_D} E(v^c_{j'}(s), G(s))}{|S_D|} \tag{11}$$

Chapter 6 describes several rank-level and measurement-level combination techniques, and also describes techniques that make use of an ensemble of audio-video based recognizers.

A/V Character Recognition Accuracy. The end-to-end A/V character recognition accuracy, α, computed on the entire test set S, with A/V combination employed on the set of ambiguous segmented character S_D and purely video based character recognition employed on the set of non-ambiguous segmented characters $S - S_D$ is given by

$$\alpha(S) = \frac{|S - S_D| \times \alpha_V(S - S_D) + |S_D| \times \alpha_Z(S_D)}{|S|} \tag{12}$$

2.3 Character Recognizer

As described earlier in this chapter, the character recognition component is expected to produce, for each input segmented character, a set of possible character recognition outputs, called the video options. This set of video options is subsequently used by the various components of the end-to-end recognition system to arrive at the final output of the audio-video based character recognition stage.

In the current implementation of the end-to-end system, we have made use of a character recognition component that is based on the optical character recognition tool called GNU Optical Character Recognition or GOCR [33]. Note that our implementation is not tied to this particular character recognizer *i.e.* GOCR. A different character recognizer that returns a set of possible character options (*i.e.* video options) along with their corresponding match scores or one that can be modified to return such an output for an input segmented character image, may also be used.

2.3.1 GOCR

GNU Optical Character Recognition or GOCR [33] is an open source OCR program developed under the GNU Public License. It converts text images into machine-readable text. It is a rule-based recognizer that was designed for printed characters and therefore, it works best when the handwritten character looks very similar to the printed equivalent.

The GOCR tool first computes measures such as the number of loops, the number of lines and the number of crossings using primitive functions such as `loop()`, `line()` and `num_cross()` and then, it makes use of a set of pre-defined rules that operate on these measures to return the recognition output. Each character has a different set of rules. The original GOCR engine is designed to return a single recognized character for every segmented character and in some cases when the specific set of rules is not satisfied for any character in the character set, the segmented character is not recognized. Also, the GOCR tool does not return any video match score corresponding to the recognized character.

2.3.2 Our Implementation

We have made two main modifications to the existing GOCR tool. The first modification is to return a set of video options instead of returning a single recognized

character or no match at all. The modified GOCR is capable of returning the complete list of possible recognized characters (*i.e.* video options) for each segmented character, even in cases when the video match score is not very high. We may later chose to prune this set to return only a small set of video options based on some conditions. The second modification is to return the video match scores corresponding to each of the video options. The video match score is meant to correspond to the recognizer's confidence that the segmented character is actually that particular video option and it is computed based on the number and the relative significance of recognition rules that have been satisfied. The current engine recognizes alphabets (both capital and small), numbers and basic arithmetic operators.

2.3.3 Performance

The character recognition accuracy of our character recognizer, which includes the GOCR tool and several modifications that we have made, has been observed to be 53.7% for one of our test data set (discussed in Section 2.5). Since GOCR was designed to recognize printed characters, it works best for those handwritten characters that look very similar to their printed equivalents. On the other hand, our character recognizer is unable to accurately recognize characters that are significantly different when handwritten using cursive style. Therefore, in our data set we attempt to write all characters in the printed style and do not use the cursive style. Example sets of characters that are commonly confused are {'g', '9' and 'q'} and {'+', 'f' and 't'}. Examples of characters for which our character recognizer has a very high recognition accuracy are 'C', 'K' and 'V', and those with low recognition accuracy are 'g', 't' and '2' since each of these character often tends to be recognized as another similar looking character ('9','+' and 'Z', in this case).

2.4 Audio Text Recognizers

In our end-to-end system we make use of two different audio text recognizers. The first one, used in the audio-video based character disambiguation stage by the synchronization component, is a commercial phonetic search tool called Nexidia [74]. Phonetic search, as provided by Nexidia, has the advantage that it does not require a grammar or a lexicon. This feature becomes important because the audio content contained in the classroom video does not necessarily follow a strict grammar, mathematical or otherwise. For character disambiguation, Nexidia is able to efficiently locate the utterances of a specific character in the audio content without the need for any grammar or restrictions on how the content was spoken by the instructor. The second audio text recognizer, used in the grammar assisted audio-video based content disambiguation stage, is implemented using the Sphinx-4 toolkit [98] and makes use of a mathematical grammar. While the Sphinx based audio text recognizer, because of its use of grammar, is able to achieve higher recognition accuracy, it also imposes restrictions on the content spoken by the instructor. The remainder of this section provides an overview of Nexidia and Sphinx-4, and our use of the two technologies.

2.4.1 Nexidia Based Audio Text Recognizer

Nexidia[1] is a commercial phonetic search software. Nexidia operates in two stages. The first stage involves converting the entire audio file to a time-aligned index of phonetic content, called the Phonetic Audio Track (PAT) file, that is accessed when searches and queries are performed. This indexing process only takes place once and there is no need to re-index. In the second stage, Nexidia uses a patented phonetic search technology to locate possible occurrences of the input search term in the PAT file that was generated. High search speeds and the ability to work with non-standard grammar patterns and different languages without prior training are some of the

[1] I would like to thank Nexidia Inc. for providing access to the software, free of cost.

advantages of this software.

Our implementation of the Nexidia based audio text recognizer consists of *search term* generation and is followed by phonetic search using Nexidia. Search term generation involves generating the audio or phonetic sequence for which we wish to find occurrences in the audio component using the Nexidia software. In our implementation we have used a character to audio search term map (shown in Table 1) that maps each character in the dictionary C (*i.e.* the set of characters that we intend to recognize) to one or more audio search terms. For example, we may allow the character `'+'` to be spoken as a `'plus'`, `'add'` or `'sum'` and each of these forms a possible search term for the character `'+'`. Nexidia allows the search term to be in the phonetic form (*e.g.* `'_p_l_ah_s'`) or regular spelling (*e.g.* `'plus'`) which it converts into the equivalent phonetic sequence. In our setup, we have configured Nexidia to use the pronunciation model and the acoustic model that correspond to the *North American Broadcast English*.

Table 1: Character to Audio Search Term Map for a few characters

Text	**Audio Search Terms**
+	*plus*, *add*, *sum*
−	*minus*, *subtract*
1	*one*
2	*two*, *square*
z	*zed*, *zee*

We have observed that the recognition accuracy of the audio text recognizer varies significantly between different search terms. For smaller search terms *i.e.* when there are less number of phonemes, we have observed that the false positives are quite high and the false negatives are low. Also search terms that have many common phonemes in a similar sequence such as `'b'`, `'c'` and `'d'` tend to be commonly confused. For larger search terms such as `'plus'`, `'seven'` and `'one'`, the number of false positives is very low.

2.4.2 Sphinx-4 Based Audio Text Recognizer

Sphinx-4 [98] is a speech recognition system written entirely in the Java programming language. It is a result of a collaboration between several academic and industry groups, including the Sphinx group at Carnegie Mellon University, Mitsubishi Electric Research Labs (MERL), Sun Microsystems Labs and Hewlett Packard (HP). The University of California at Santa Cruz (UCSC) and the Massachusetts Institute of Technology (MIT) have also participated actively in the development of the recognition system. Sphinx-4 has a highly flexible and modular design [99]. The modularity allows one to easily modify and experiment with a specific module of the Sphinx-4 system without having to modify other portions of the system. Sphinx-4 consists of three primary modules: *FrontEnd*, the *Linguist*, and the *Decoder*. The *FrontEnd* takes one or more input signals, processes them and generates a sequence of *Features*. The *Linguist* consumes a standard language model, along with pronunciation information from a *Dictionary* and structural information from one or more sets of *AcousticModels*, to generate a *SearchGraph*. A *SearchManager*, which is a part of the *Decoder*, uses the *Features* generated by the *FrontEnd* and the *SearchGraph* generated by the *Linguist* to perform the actual decoding and produce the *Results*.

A key capability provided by the Sphinx-4 framework is its flexibility, which allows one to programmatically influence the recognition process at any time before or during the recognition process. In our implementation, we exploit this flexibility of the Sphinx-4 framework and the ability to programmatically issue controls to replace, during the recognition process, the grammar being used by the speech recognizer. More details about the two grammars, the *base mathematical speech grammar* and the *constrained speech grammar*, that are used as part of our grammar assisted audio-video based handwritten mathematical content recognition technique will be provided in Chapter 7.

Table 2: Details of the training and evaluation data sets

Data Set Name	Number of Video Segments	Number of Characters
DS-TRAIN-1	65	3058
DS-TEST-1	70	3630
DS-TRAIN-2	30	1356
DS-TEST-2	35	1440

2.5 Data Set

We now provide a description of the training and the test data sets that are used for training and evaluation of the techniques proposed in this dissertation. The recording equipment used for capturing the videos for our data set consisted of a commercially available off-the-shelf video camera (Sanyo VPC-HD1A 720p High-Definition Digital Media Camera [90]) and a wired microphone. The videos were recorded in a classroom-like setting, and capture the content being written on the whiteboard and the content being spoken by the instructor. The camera is set up to capture videos with a resolution of 1280 x 720 pixels at 30 frames per second. These videos have been recorded by two subjects (referred to as instructors). While the same whiteboard is used for all the recordings, the illumination and the background for such recordings have minor variations between the recording sessions. The use of an off-the-shelf microphone and the intermittent sounds associated with the movement of the pen on the whiteboard introduce noise-like artifacts in the audio track.

We have broadly organized the recorded videos into four data sets, two for training and two for evaluation. The two data sets corresponding to subject 1 will be referred to as `DS-TRAIN-1` and `DS-TEST-1`, while the data sets corresponding to subject 2 are referred to as `DS-TRAIN-2` and `DS-TEST-2`. Note that each recorded video from a given data set can have multiple *video segments*, each of which starts with a blank

whiteboard and ends with the complete erasure of the content written on the whiteboard during the video segment. The details of the two data sets in terms of video segments and number of characters are provided in Table 2. The video component of the data set is encoded as an MPEG-4 stream and the audio component is a PCM audio with a bit rate of 1536 kbps and an audio sample rate of 48 kHz. A sample of this data set is available online [18], along with the instructions for obtaining the complete data set.

To enable consistent comparison and to allow the reader to easily examine and evaluate the improvements achieved by each of the proposed techniques, we make use of the test data set `DS-TEST-1` across all the components of the end-to-end recognition system. The results from the second test data set `DS-TEST-2` are presented for the audio-video combination component only. The training data sets are used by several techniques in the proposed audio-video recognition system, which include the mapping technique presented in Section 4.2.1, the character-specific thresholding technique presented in Section 4.2.2.2 and the character-specific weighted combination technique presented in Section 6.3.3.

One should note that although we will be referring to these videos as *classroom videos*, the audio-video based techniques presented in this dissertation are not limited to videos recorded in a classroom and some techniques may even be applicable to entirely different applications.

CHAPTER III

VIDEO PREPROCESSING

3.1 Introduction

The video preprocessing stage is the first stage of the end-to-end audio-video based recognition system and is responsible for extracting handwritten characters and associated features, like the location and the timestamp information, from the video component of the classroom video. In the context of the end-to-end system, shown in Figure 5, the video-preprocessing stage immediately precedes the character recognizer and the two together constitute the video text recognizer. Given an input video, the video preprocessing stage starts by determining the *region of interest* for the video, which is the area in a given video frame that can possibly contain the handwritten characters and corresponds to the white region of the whiteboard. Next, the video is partitioned into multiple time segments, called *durations of interest*, each roughly separated from the other by a complete erasure of the whiteboard. For each duration of interest, the preprocessing stage finds a frame which contains all the characters written before a complete erasure and does not contain the instructor within the

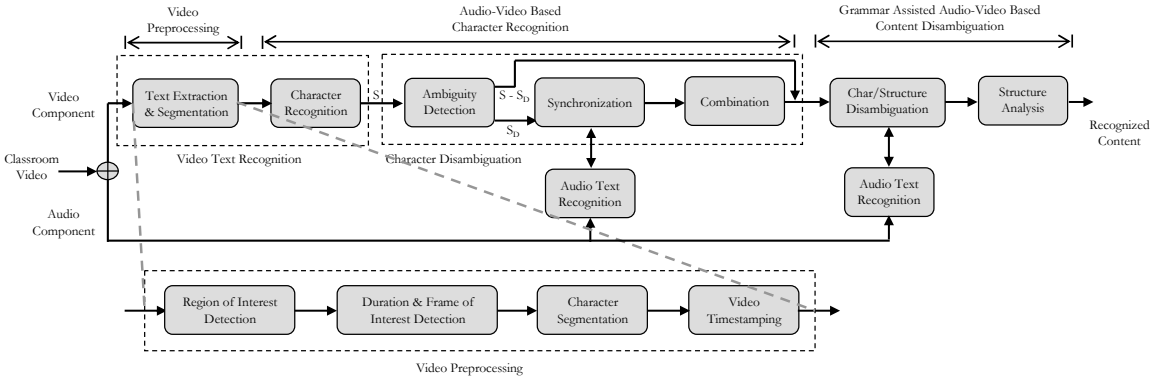


Figure 5: Stages involved in video preprocessing

region of interest. This frame is called the *frame of interest*. Thereafter, for each duration of interest, the video preprocessing stage operates on the corresponding frame of interest and outputs a *set of segmented characters* written by the instructor along with the *location coordinates* and the *video timestamp*.

Figure 5 shows a schematic representation of our implementation of the video preprocessing stage, which is an assembly of components that extend and/or utilize several basic video and image processing techniques. Specifically, the preprocessing stage consists of four components, which correspond to the region of interest detection, the duration of interest and frame of interest detection, character segmentation and video timestamping. We start by describing the assumptions that are made by the preprocessing stage with regards to the recording setup and the input classroom video and also provide details of a connected component analysis technique, an implementation of which is used across several components of the preprocessing stage. This is followed by a detailed description of the components that constitute the preprocessing stage along with sample outputs for each of the components. Finally, we present a brief evaluation of the preprocessing stage and specify the values assigned to the parameters used in our implementation of the various components of preprocessing.

3.1.1 Assumptions

In order to automate various aspects (*e.g.* region of interest detection, duration of interest detection, etc.) of the preprocessing stage, we make certain assumptions about (1) the setup that is used for recording the videos and (2) the content of the input classroom videos. Although these assumptions are not very restrictive, they help us in significantly reducing the complexity and improving the performance of the preprocessing stage when compared to a preprocessing stage that operates without making any of the assumptions listed below.

Assumptions about the recording setup:

- The entire frame of the whiteboard is completely inside the video frame.
- The whiteboard is the largest object in the video frame.
- The edge of the whiteboard is significantly darker than the whiteboard.

Assumptions about the content of the input classroom videos:

- Every video that is recorded on a new setup has at least one known *calibration frame*, which is a video frame with a clean whiteboard and without the instructor. This frame is usually amongst the first few frames of the video.
- No text is written very close (within a few pixels) to the edge of the whiteboard, which is usually the case with any classroom video.
- All the contents of the whiteboard are erased well before the instructor starts to write again.
- The preprocessing stage is not designed to handle partial erasures of the whiteboard. However, it is capable of handling erasures and substitutions of a few characters except when the partial erasure or correction happens after the detected frame of interest *i.e.* the frame which has the maximum number of connected components and does not have the instructor obstructing any part of the region of interest.
- The instructor, after finishing writing on the whiteboard, briefly steps away from the whiteboard (*i.e.* the region of interest) so that the region of interest is not obstructed and there is no shadow being cast by the instructor on the whiteboard.

Note that these assumptions also help us in improving the quality of the preprocessing output, thereby improving the recognition accuracy of the end-to-end system.

If needed, the preprocessing stage can be altered to accommodate any changes in the listed assumptions. In scenarios where the recording setup is fixed and/or there are more restrictions imposed on the instructor, the preprocessing stage can be further simplified and tuned for better output.

3.1.2 Connected Component (CC) Analysis

Connected component (CC) analysis is the process of grouping pixels into components based on pixel connectivity. This process scans the image pixel by pixel and returns the set of connected components where each connected component is made up of pixels that have the same intensity value and are also connected to each other by a defined connectivity measure. Connected component analysis can be performed using different connectivity measures such as 4-connectivity or 8-connectivity. More details about connected component analysis can be found in [34]. The implementation of the component labeling algorithm used in our preprocessing stage utilizes a contour tracing technique [15] to detect all connected components in a given image, and is based on the `OpenCV` [76] library. Connected component analysis is used by several components of the preprocessing stage, *e.g.* to locate the area of the whiteboard inside the whiteboard frame by detecting white connected components (those pixels with an intensity of 1) in a video frame and to locate characters in a video frame by detecting black connected component (those pixels with an intensity of 0). The connected component analysis procedure returns the set of connected components with the pixel information and the coordinates of the bounding box for each connected component.

3.2 Region of Interest Detection

A classroom video, besides capturing the content that is handwritten on the whiteboard, also captures the associated background and foreground objects and other artifacts. In the context of our system, we consider the background to be the frame

Algorithm 1: Region of Interest Detection

input : Calibration Frame $\mathtt{F}_{\mathtt{calib}}$, Threshold $\tau_{\mathtt{ROIBinarize}}$, Dilations `d`, Erosions `e`
output: Region of Interest Filter $\mathtt{F}_{\mathtt{ROI}}$

```
1 F_Gray ← grayScale(F_calib);
2 F_Thresholded ← binarize(F_Gray, τ_ROIBinarize);
3 blackCCs ← detectBlackCCs(F_Thresholded);
4 whiteCCs ← detectWhiteCCs(F_Thresholded);
5 biggestBlackCC ← findMaxSizeCC(blackCCs);
6 ROICC ← findMaxSizeCC(whiteCCs contained-in biggestBlackCC);
7 F_ROI ← createROIFilter(ROICC, coordinates(F_calib));
8 dilate(F_ROI,d);
9 erode(F_ROI,e);
```

of the whiteboard and the area of the video frame outside the frame of the whiteboard. The foreground, in our system, is the instructor and the shadows cast by him/her. It is obvious that the presence of such objects and artifacts in the background and the foreground complicates the detection of handwritten characters from the whiteboard. Towards this end, the *region of interest* identifies the white region inside the whiteboard frame where we allow characters to be written and attempts to solve a part of this problem by using a *region of interest filter*, $\mathtt{F}_{\mathtt{ROI}}$, to remove the background objects and artifacts such as the edge of the whiteboard and anything else that is outside of the whiteboard frame.

3.2.1 Algorithm

We now describe the procedure for determining the region of interest (ROI) filter, $\mathtt{F}_{\mathtt{ROI}}$. This algorithm, besides using the calibration frame as input, also makes use of certain parameters to determine the ROI filter. A concise listing of the algorithm in pseudo-code format is included as Algorithm 1.

1. ***Binarizing the calibration frame:*** The algorithm starts with a specific calibration frame, which is converted to a grayscale image and subsequently converted to a binary image by using the threshold $\tau_{\mathtt{ROIBinarize}}$, where all pixel values above or equal to $\tau_{\mathtt{ROIBinarize}}$ are assigned a value of 1 (*i.e.* white) and

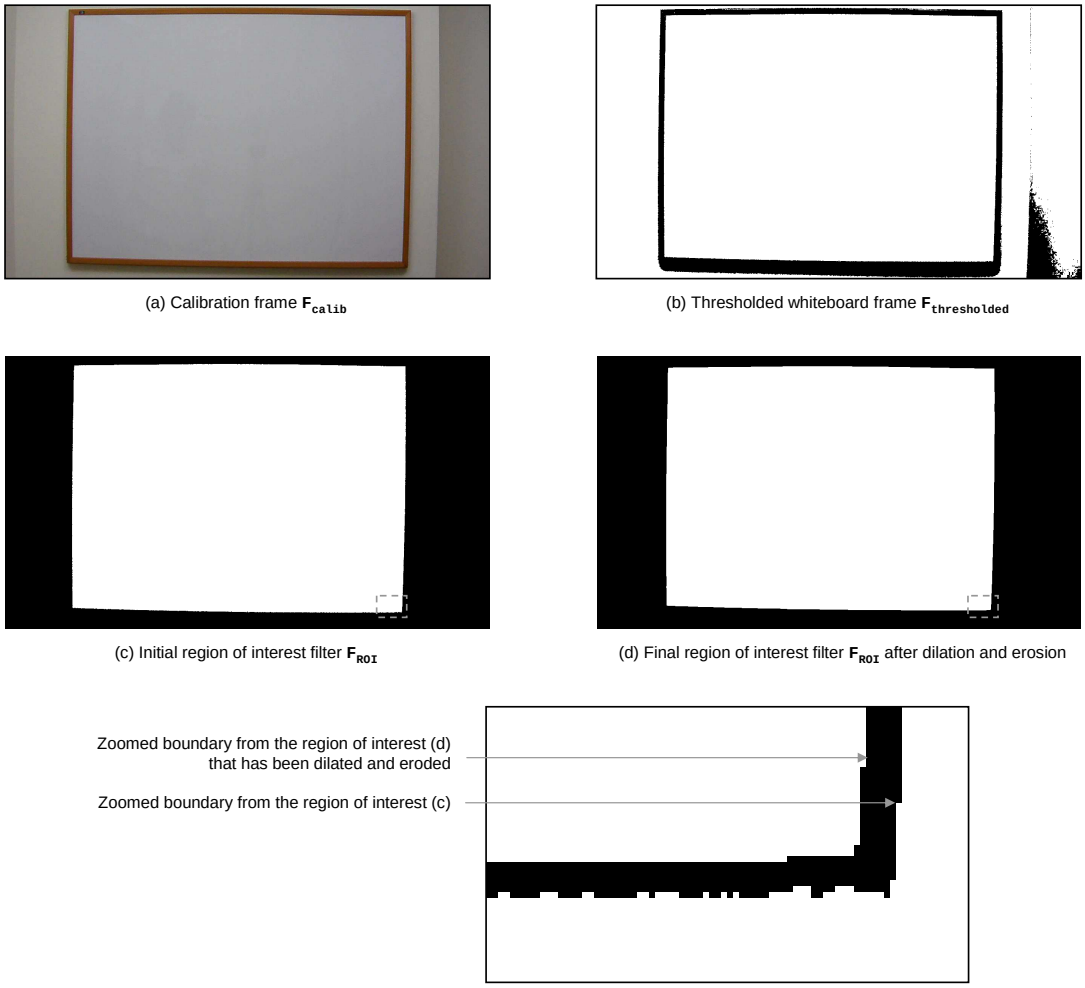
others are assigned a value of 0 (*i.e.* black).

2. ***Detecting the black and white connected components:*** We make use of the connected component analysis procedure (described in Section 3.1.2) to find a list of all the black connected components and the white connected components in the binarized image.

3. ***Detecting the whiteboard frame and the region of interest:*** The whiteboard frame is assumed to be the largest black connected component, and the largest white connected component that is contained within this whiteboard frame is considered to be the writing area, called the region of interest. Here, the largest connected component is determined based on the area of the bounding box and not the number of pixels.

4. ***Creating the region of interest filter:*** We create the region of interest filter with a size equal to that of the video frame (here, we use the calibration frame) and fill it with white pixels in the region of interest and with black pixels in the remaining area.

5. ***Removing unwanted artifacts from the region of interest filter:*** We perform `d` iterations of dilation to remove any small-sized black artifacts that lie in the ROI, possibly caused by uneven lighting and/or handwritten content that may have not been erased completely. Such artifacts may result in the loss of information at these pixels when the ROI filter is subsequently applied to a video frame. Dilation is followed by `e` iterations of erosion to reduce the size of the region of interest that was increased due to dilation. We ensure that $\texttt{e} > \texttt{d}$, and as a result the size of the ROI is reduced by $(\texttt{e} - \texttt{d})$ pixels at each edge of whiteboard, which eliminates any random artifacts that may appear near the edge of the whiteboard frame due to camera jitter or slight movement of the whiteboard upon use by the instructor.



(a) Calibration frame $\mathbf{F}_{\texttt{calib}}$

(b) Thresholded whiteboard frame $\mathbf{F}_{\texttt{thresholded}}$

(c) Initial region of interest filter $\mathbf{F}_{\texttt{ROI}}$

(d) Final region of interest filter $\mathbf{F}_{\texttt{ROI}}$ after dilation and erosion

(e) Shows the pixel-wise NXOR between the selected region from (c) and (d). Notice that the boundary of region in (d) is dilated inwards as opposed to the boundary of region in (c)

Figure 6: Determining the region of interest filter

3.2.2 Example

The intermediate output from the various steps of the region of interest detection algorithm is shown in Figure 6. Given a calibration frame (shown in Figure 6(a)) from an input classroom video, the output image resulting from binarization is shown in Figure 6(a). The connected component analysis is performed on the binarized image, and the connected component corresponding to ROI is then detected using the above mentioned algorithm. An initial ROI filter with the same dimensions as the calibration frame and containing only the ROI connected component is then created.

This is shown as Figure 6(c). Any small artifacts in the initial ROI filter are then removed by performing specified number of dilation and erosion iterations, and the resulting ROI filter is shown in Figure 6(d). As shown in Figure 6(e), the final ROI area is a few pixels smaller than the initial ROI on each side, which helps to remove the artifacts caused by the movement of the whiteboard frame when the instructor writes.

Algorithm 2: Duration of Interest & Frame of Interest Detection

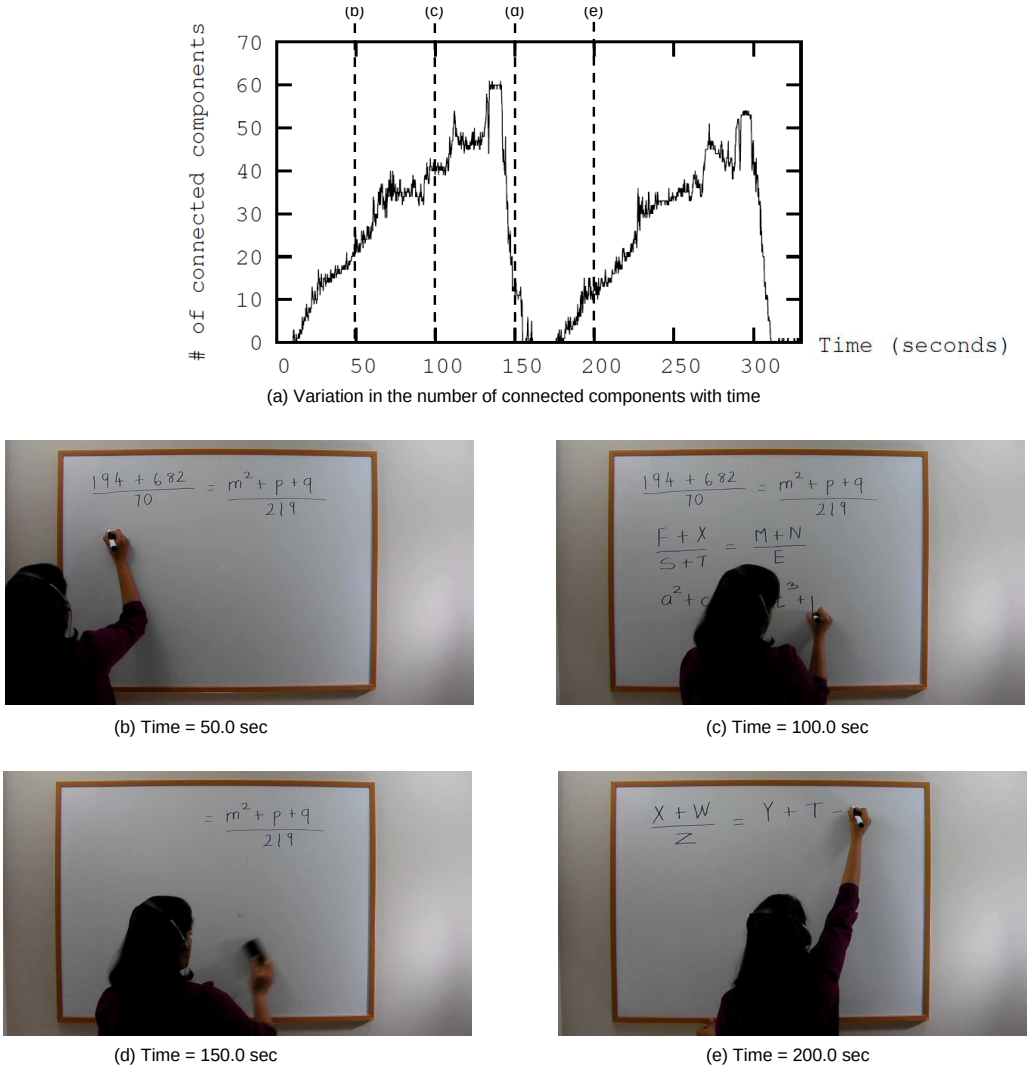
input : Classroom Video $\texttt{V}$, Calibration Frame $\texttt{F}_{\texttt{calib}}$, Region of Interest Filter $\texttt{F}_{\texttt{ROI}}$, Sampling Increment $\texttt{N}_{\texttt{sample}}$, Thresholds $\tau_{\texttt{charBinarize}}$, $\tau_{\texttt{minPixels}}$, $\tau_{\texttt{maxPixels}}$, $\tau_{\texttt{DOIDetect}}$

output: Duration of Interests `finalDOIs`, Frame of Interests `finalFOIs`

```
globals: occlusionPlot, componentCountPlot;
for i ← 1 to numFrames(V) increment-by N_sample do
    F_Gray ← grayScale(frameAtIndex(V, i));
    F_Diff ← frameDifference(F_Gray, grayScale(F_calib));
    F_Thresholded ← binarize(F_Diff, τ_charBinarize);
    F_ThresholdedInverted ← invertBlackWhite(F_Thresholded);
    blackCCs ← detectBlackCCs(F_ThresholdedInverted);
    bigBlackCCs ← detectBigBlackCCs(BlackCCs, τ_maxPixels);
    addToOcclusionPlot(isROIOccluded(F_ROI, bigBlackCCs));
    prunedBlackCCs ← pruneCCs(blackCCs, τ_minPixels, τ_maxPixels);
    prunedROIBlackCCs ← ROIFiltering(prunedBlackCCs, F_ROI);
    addToComponentCountPlot(count(prunedROIBlackCCs));
initialDOIs ← generateInitialDOIs(τ_DOIDetect);
initialFOIs ← generateInitialFOIs(initialDOIs);
[finalDOIs, finalFOIs] ← mergeDOIsAndSelectFOIs(initialDOIs, initialFOIs);
```

3.3 Duration of Interest and Frame of Interest Detection

We define the *duration of interest* to be the period between two complete erasures of the whiteboard. A classroom video may have several durations of interest based on how many times the whiteboard has been completely erased. The duration of interest proves to be useful for subsequent video preprocessing steps such as determining the frame of interest, and for enforcing an upper limit on the time window within which the searches for audio content related to handwritten content contained in the



(a) Variation in the number of connected components with time

(b) Time = 50.0 sec

(c) Time = 100.0 sec

(d) Time = 150.0 sec

(e) Time = 200.0 sec

Figure 7: Showing the character count plot with sample frames for a classroom video

duration of interest should be performed.

The *frame of interest* is defined as the frame between two complete erasures *i.e.* within a duration of interest, which has the maximum number of connected components within the region of interest and where the instructor and his/her shadow are not occluding any part of the region of interest. For each duration of interest, the frame of interest is identified and each connected component (after basic pruning) contained in the region of interest of this frame is considered to be a character. This frame is assumed to contain all the characters handwritten in the corresponding duration of interest.

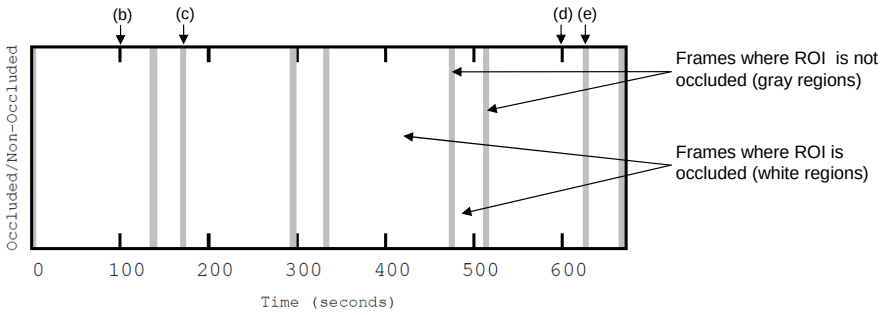
3.3.1 Algorithm

Given a classroom video, the procedure for determining the durations of interest and the corresponding frames of interest is described in the algorithm below. A concise listing of the algorithm in pseudo-code format is included as Algorithm 2.

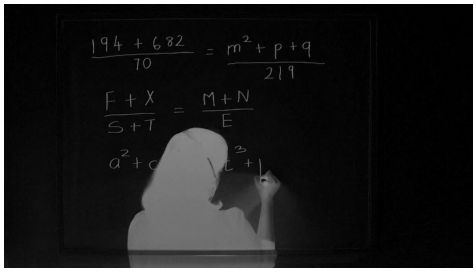
1. ***Frame differencing and binarization:*** The video sequence is sub-sampled by a factor of $\mathtt{N_{sample}}$ *i.e.* the first frame out of every $\mathtt{N_{sample}}$ frames in selected. For each frame in the sub-sampled sequence, frame differencing is done between that frame and the calibration frame. To some extent, this addresses the fact that illumination conditions are not uniform throughout the whiteboard. The frame differencing output for each sub-sampled frame is binarized into a black and white image such that pixels with intensity greater than or equal to $\tau_{\mathtt{charBinarize}}$ become white and those less than $\tau_{\mathtt{charBinarize}}$ become black.

2. ***Connected component analysis:*** Using connected component analysis, the set of black connected components is detected. Pruning thresholds $\tau_{\mathtt{minPixels}}$ and $\tau_{\mathtt{maxPixels}}$ are employed to remove those connected components with less than $\tau_{\mathtt{minPixels}}$ black pixels, which in most cases correspond to small noise artifacts and those with more than $\tau_{\mathtt{maxPixels}}$ pixels, which generally correspond to artifacts caused by the instructor and his/her shadow.

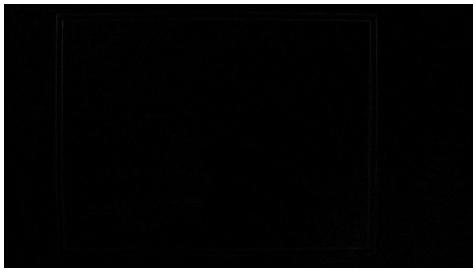
3. ***Generating the component count plot and occlusion plot:*** The component count plot (example shown in Figure 7) depicts the variation in the number of connected components (after pruning) that fall completely within the ROI (determined using the ROI filter) with time. The occlusion plot (example shown in Figure 8) shows, by means of gray bars, the durations when the ROI is not occluded by the instructor or his/her shadow. The time corresponding to the left edge of the bar indicates the start of the time in the video when the ROI stops being occluded and the time corresponding to the right edge of the bar



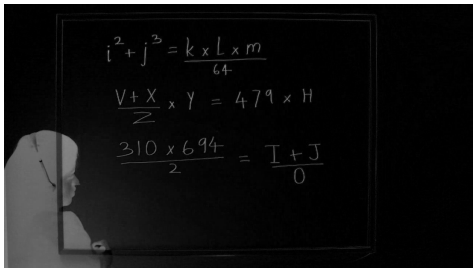
(a) Occlusion plot



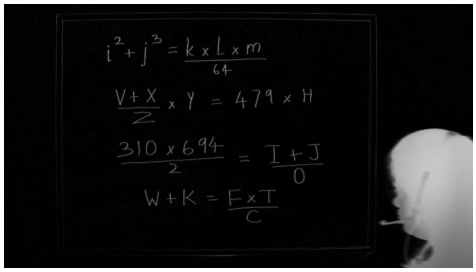
(b) time = 100.0 seconds (ROI occluded)



(c) time = 170.0 seconds (ROI non-occluded – Not FOI)



(d) time = 600.0 seconds (ROI occluded)



(e) time = 628.8 seconds (ROI non-occluded - FOI)

Figure 8: Showing the occlusion plot with sample occluded and non-occluded regions of interest for a classroom video

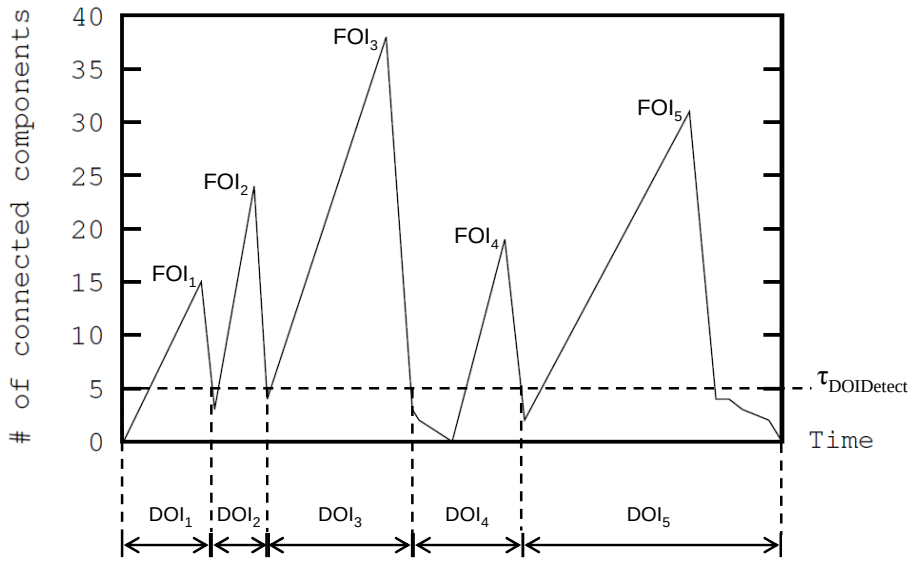
indicates the time in the video when the ROI is occluded again. Occlusion of the ROI is considered to take place when a connected component with greater than $\tau_{\texttt{maxPixels}}$ pixels (*i.e.* the instructor or his/her shadow) overlaps with any portion of the ROI.

4. ***Determining the initial durations of interest:*** The first frame of the video file is considered to be the start of the first duration of interest DOI_1 and the last frame of the video file is the end of the last DOI. As expected the end of a DOI also forms the beginning of the next DOI. Every time the component

count plot transitions from above the value $\tau_{\texttt{DOIDetect}}$ to a value below or equal to it, that frame of transition is considered to be the end of the previous DOI and the beginning of the next DOI. Figure 9(a) illustrates the correspondence between the initial DOIs and the component count plot. Note that the figure is only for illustrating the working of the algorithm and does not correspond to a real input video from the classroom.

5. ***Determining the initial frames of interest:*** In each DOI, amongst the frames without occlusion, the frame containing the highest number of connected components is considered to be the corresponding frame of interest FOI. If there is no frame without occlusion in a given DOI (*i.e.* a frame of interest does not exist), then this DOI is merged with the next DOI and the FOI is determined for the new merged DOI. Figure 9(a) shows the initial FOIs.

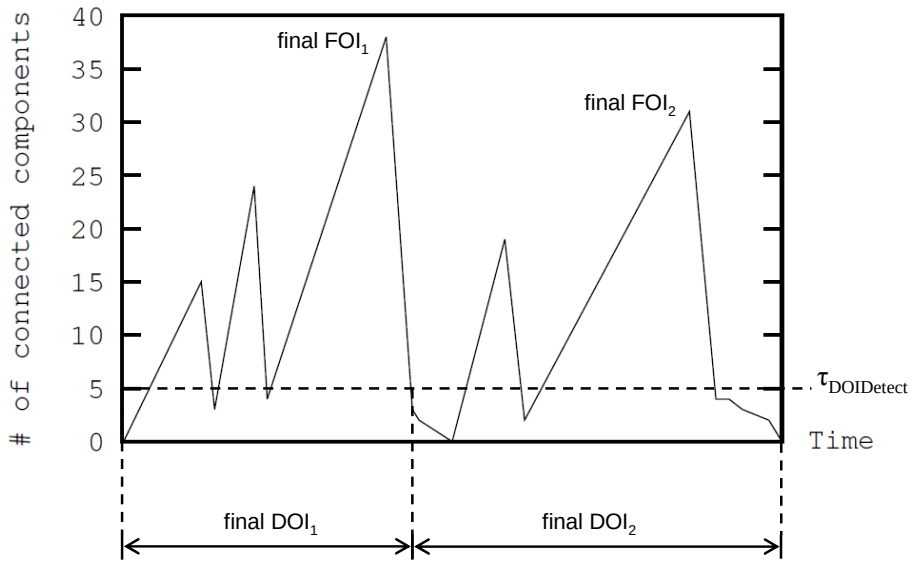
6. ***Merging the durations of interest and selecting the final frames of interest:*** In some cases, due to occlusions caused by the instructor, a single DOI may be recognized as two separate DOIs, each with its own FOI. To handle such cases, we compare the connected components from the FOIs of two adjacent DOIs for similarity. If there is a significantly large fraction of connected components between the two FOIs that are similar then we proceed with merging the two corresponding DOIs. The procedure for determining if two FOIs have a significant overlap in terms of the corresponding connected components is described in Section 3.3.2, we refer to the procedure as `FOI_Match`. For the set of DOIs determined from an input video, the `FOI_Match` function is evaluated for the first and second FOIs. If the match condition is satisfied *i.e.* the `FOI_Match` function returns the value `TRUE`, then their corresponding DOIs are merged together to form a single DOI and between the two FOIs, the one which has the higher number of connected components is selected to be the FOI of



(a) Initial DOIs and initial FOIs

FOI_Match(FOI$_1$, FOI$_2$) = TRUE => merge DOIs
FOI_Match(FOI$_2$, FOI$_3$) = TRUE => merge DOIs
FOI_Match(FOI$_3$, FOI$_4$) = FALSE => do not merge DOIs
FOI_Match(FOI$_4$, FOI$_5$) = TRUE => merge DOIs

(b) Evaluating **FOI_Match** condition to merge initial DOIs



(c) Final DOIs (by merging initial DOIs) and final FOIs (selected from initial FOIs)

Figure 9: Working of the duration and frame of interest detection algorithm

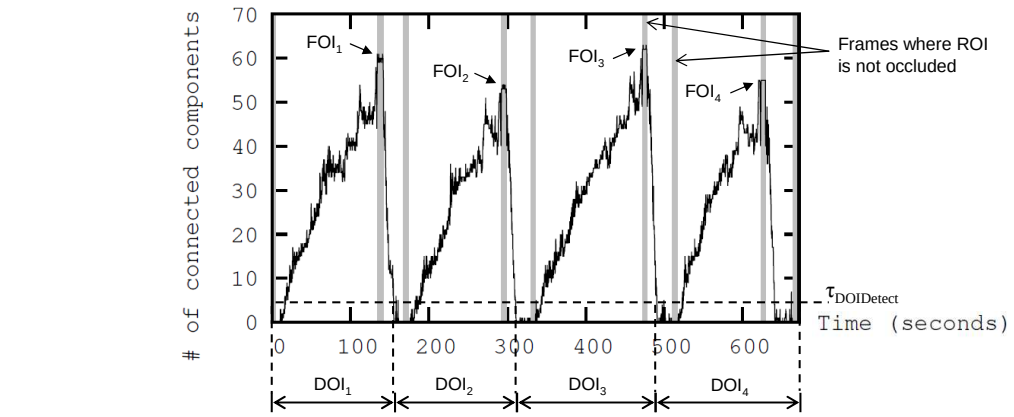
the new merged DOI. Now, this new FOI is compared with the third FOI and so on. If the match condition described above is not satisfied for the first and second FOIs *i.e.* the `FOI_Match` function returns the value `FALSE`, then the two FOI frames are considered to belong to different DOIs and then we start comparing the second and third FOIs to check if they belong to the same DOI. This process is continued until we reach the end of of all detected DOIs. Figure 9(b) illustrates the process used for merging the DOIs and the FOIs found as part of Figure 9(a). Shown in Figure 9(c) are the DOIs and FOIs that finally remain are the real DOIs and FOIs that are used for further preprocessing.

3.3.2 `FOI_Match` Function

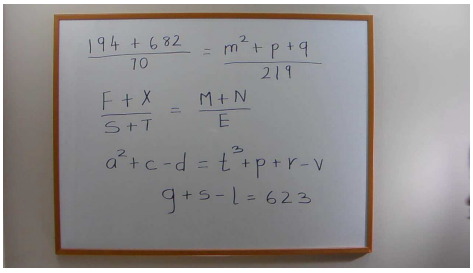
The `FOI_Match` function is used to determine whether two given FOIs belong to the same DOI or not. The function determines the number of connected components that are common between the frames and if the ratio of this number and the minimum of the number of connected components in each of the two input FOIs is higher than or equal to $\tau_{\texttt{FOIMatch}}$, then the FOIs are considered to be matched *i.e.* the two FOIs belong to the same DOI. When the FOIs match, then the `FOI_Match` function returns the value `TRUE`, else the value `FALSE`. The process for determining the common connected components between the two FOIs relies on a function, `CC_Match` (described in Section 3.5.2), to determine if two connected components are matched or not.

3.3.3 Example

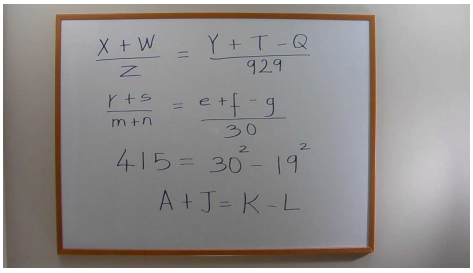
A sample component count plot that is generated for determining the DOIs and FOIs for a given input video is shown in Figure 7(a) along with four sample frames from the plot (shown in Figure 7(b), (c), (d) and (e)). Another plot that is used by the DOI and FOI detection algorithm is the occlusion plot and a sample of this for the same input video is shown in Figure 8 with sample occluded and non-occluded frames. The white parts of this plot correspond to frames where the ROI was occluded by some



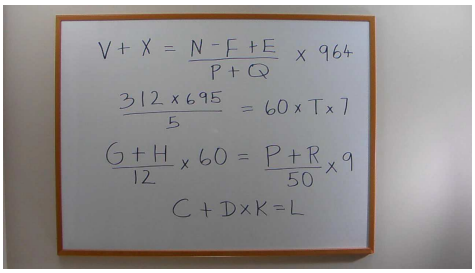
(a) Character count plot and occlusion plot



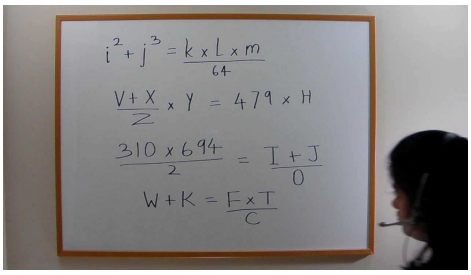
(b) Time = 140.8 sec (FOI_1)



(c) Time = 297.0 sec (FOI_2)



(d) Time = 476.8 sec (FOI_3)



(e) Time = 628.8 sec (FOI_4)

Figure 10: Showing the automatically detected durations and frames of interest for a classroom video

part of the instructor or his/her shadow such as in Figure 8(b) and (d) and the gray regions correspond to frames where the ROI was non-occluded such as in Figure 8(c) and (e).

Figure 10 shows the DOIs and the FOIs automatically detected by the implementation of the algorithm outlined above for an input classroom video. The input classroom video has four DOIs and four FOIs (shown as Figure 10(b),(c),(d) and(e)) that were correctly recognized by making use of the algorithm. Each of these FOIs contains all the characters that were written within the corresponding DOI and the

ROI is also not occluded by the instructor or his/her shadow. In Figure 10(d), we see that the instructor is within the video frame but not occluding the ROI. Figure 10(a) shows the component count plot and the occlusion plot for the input video with the white parts corresponding to occluded frames and the gray regions corresponding to non-occluded frames. $\tau_{DOIDetect}$ is the threshold used for detecting initial DOIs.

3.4 Character Segmentation

Character segmentation is a well known problem in the domain of printed and handwritten text recognition. It involves processing the input image (in this case, the input video) to return a set of sub-images, each of which contains a separate character. In the case of handwritten cursive text where the characters do not occur as disjoint components, some very sophisticated character segmentation techniques may be required and in some cases, the recognition is done without a separate segmentation stage. In case of handwritten mathematical equations, however, the majority of characters (*e.g.* variables, symbols and numerals) appear as disconnected components with the exception of function names (*e.g.* `sin`, `log`, `arg`, etc.) that may sometimes be written as cursive text.

In our implementation of the character segmentation algorithm, it was assumed that the characters are written using dark ink on a clean white background, and that the individual characters are distinct. The data set that was for training and testing includes only basic algebraic expressions which are composed of numerals, the English alphabet (both capital and small), basic algebraic operators and the decimal point. Although we currently do not use functions such as sin, cos and log, we can chose to include them with the assumption that such function names are written as disconnected characters. This will require no change in the segmentation and the basic character recognition stage but would require an additional check for such sequences of alphabets after the character recognizer. A post-processing step allows us to handle

Algorithm 3: Character Segmentation

input : Frame of Interest FOI, Calibration Frame $\mathtt{F}_{\mathtt{calib}}$, Region of Interest Filter $\mathtt{F}_{\mathtt{ROI}}$, Thresholds $\tau_{\mathtt{charBinarize}}$, $\tau_{\mathtt{minPixels}}$, $\tau_{\mathtt{maxPixels}}$

output: Segmented Characters segChars

```
1  F_Diff ← frameDifference(grayScale(FOI), grayScale(F_calib));
2  F_Thresholded ← binarize(F_Diff, τ_charBinarize);
3  F_ThresholdedROI ← ROIFiltering(F_Thresholded, F_ROI);
4  F_ThresholdedROIInverted ← invertBlackWhite(F_ThresholdedROI);
5  blackCCs ← detectBlackCCs(F_ThresholdedROIInverted);
6  prunedBlackCCs ← pruneCCs(blackCCs, τ_minPixels, τ_maxPixels);
7  for i ← 1 to count(prunedBlackCCs) do
8      segChars[i].img ← createImage(boundingBox(prunedBlackCCs[i]));
9      segChars[i].img ← fillWhitePixels(segChars[i].img, FULL);
10     segChars[i].img ← fillBlackPixels(segChars[i].img, prunedBlackCCs[i]);
11     segChars[i].img ← padWithWhitePixels(segChars[i].img);
12     segChars[i].location ← getCoordinates(prunedBlackCCs[i]);
```

characters in our dataset that do not appear as a single connected component *e.g.* 'i', '=', etc.

3.4.1 Algorithm

To identify characters that are handwritten during a given duration of interest, we use the corresponding frame of interest as input to the segmentation stage. The frame of interest, by definition, is free of any occlusions or shadows caused by the instructor and has the highest number of connected components, each of which is a potential character. We make use of the procedure outlined below to extract the segmented character's image and location information. One should note that the initial steps of the segmentation algorithm are similar to those used in the frame of interest detection algorithm, but in our implementation, we chose not to store the intermediate outputs of the FOI detection algorithm and prefer to run these steps once again for the detected frame of interest. The decision to not store the intermediate results and instead repeat some of the preprocessing steps was based on the fact that the size of intermediate output can be prohibitively large even for moderate sized videos.

1. ***Frame differencing, binarization and ROI filtering:*** We make use of

absolute frame differencing between the grayscale FOI and the grayscale calibration frame to highlight the changes that have occurred between the two frames and eliminate artifacts related to non-uniform illumination. The output image has a black background with the handwritten text, the instructor (in some cases, but without obstructing any part of the region of interest) and artifacts along the edge of the whiteboard in white. To further enhance the distinction between the handwritten characters and the background, we perform binarization where any pixels with intensity above the threshold value $\tau_{\texttt{binarization}}$ become white while others become black. Next, we apply the ROI filter to this image to eliminate the instructor and any noise or artifacts that appear outside the region of interest. The black and white regions of this image are then inverted such that the text is black and the background is white.

2. ***Connected component analysis:*** Using connected component analysis, the set of black connected components is detected. The pruning threshold $\tau_{\texttt{minPixels}}$ is employed to mark those connected components with less than $\tau_{\texttt{minPixels}}$ black pixels as noise connected components. The other threshold $\tau_{\texttt{maxPixels}}$ that we use to prune away the extremely large connected components is not necessary here as the FOI does not have the instructor within the ROI and the whiteboard frame has also been removed using the ROI filter.

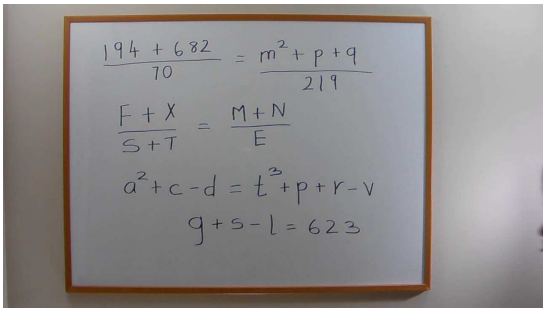
3. ***Generating segmented character elements with images and location:*** Each black connected component that remains after pruning is considered to be a segmented character and for the purpose of recognition the character must be extracted as a separate image. If we clip the black and white FOI image into sub-images based on the bounding boxes of each of the black connected components that correspond to characters and generate the respective segmented character images, then these sub-images may also include black pixels corresponding to

noise and parts of the neighboring characters. Therefore, for each character, we first create a character image with a size equal to the bounding box of the connected component, fill it with white pixels and then insert the black pixels corresponding to the connected component. By doing this we avoid including noise and also pixels that correspond to a different character that may also be within the bounding box of this connected component (examples shown in Figure 12). After this, each character image is padded by a few white pixels in each direction to accommodate for characters like the decimal point or a perfectly horizontal minus sign or division sign, which may otherwise, generate a segmented image which is completely black and cause our character recognizer to fail. The four location coordinates which correspond to the left-most, top-most, right-most, and bottom-most pixels of the connected component are also returned along with the segmented character's image.

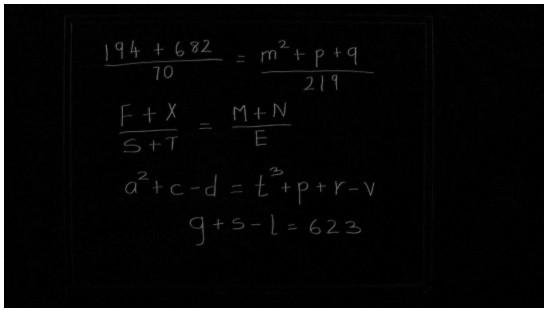
3.4.2 Example

Figure 11 shows the intermediate images from the various steps of the character segmentation algorithm. Figure 11(a) shows the original FOI. Figure 11(b) shows the image that results after differencing the grayscale FOI image with the grayscale calibration frame image. The resulting image is binarized and the output is shown in Figure 11(c). Notice the faint white artifacts that appear near the location of the edge of the whiteboard frame. The ROI filter is then applied to the binarized image and the black and white colors are inverted. The resulting image is shown in Figure 11(d). Notice that the image does not have the artifacts caused by the edge of the whiteboard. The connected components detected by the connected component analysis technique are shown in Figure 11(e), and finally Figure 11(f) shows a few sample segmented characters along with their location coordinates.

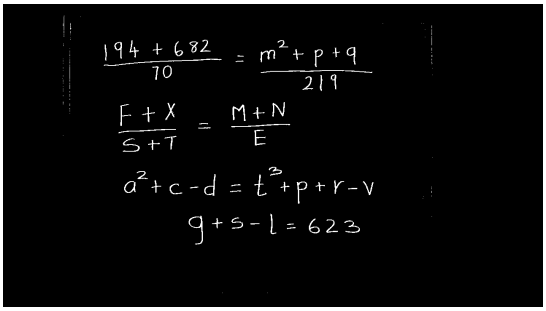
The ability of our algorithm to correctly extract character images in the case of



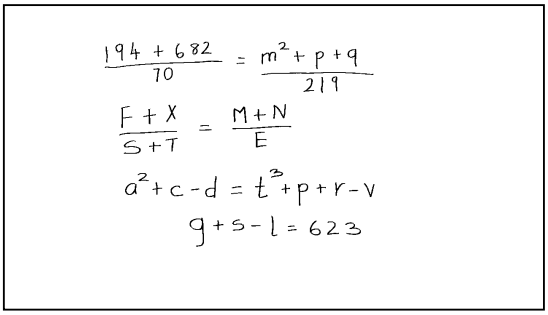
(a) Frame at t=140.8 sec (FOI)



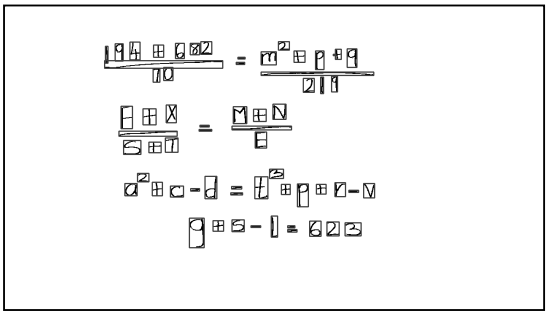
(b) Image after frame differencing



(c) Image after binarization



(d) Image after ROI filtering and inversion



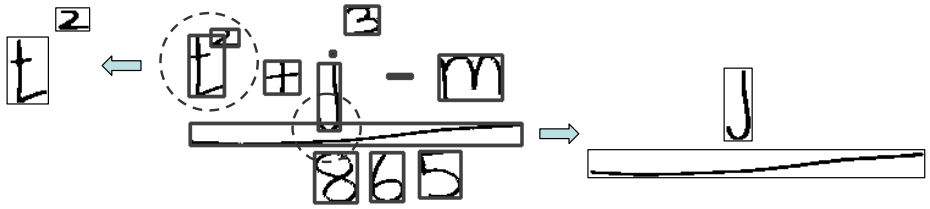
(e) Image after connected component analysis

Image	Location Coordinates
	(241, 96, 245,134)
	(264, 90, 281, 129)
	(300, 86, 321, 126)
	(355, 93, 377, 120)
	(407, 89, 426, 123)
	(442, 89, 462, 114)
	(467, 85, 491, 113)

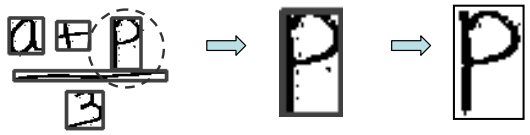
(f) Sample segmented characters

Figure 11: Images at different steps of the character segmentation procedure

overlapping bounding boxes and/or noise is shown in Figure 12. Figure 12(a) shows a small segment of the FOI where the connected components for two sets of characters (‘t’, ‘2’) and (‘j’, ‘divide-by’) overlap with each other. If the thresholded FOI is clipped based on the bounding boxes to generate the segmented character’s image, then a part of the character ‘2’ would be in the image of the segmented character ‘t’. The output of our character segmentation algorithm was able to correctly segment the characters without this problem and the output is also shown in the same figure. Figure 12(b) shows a scenario where the bounding box containing the segmented



(a) Character segmentation algorithm handles characters with overlapping bounding boxes



(b) Character segmentation algorithm removes noise artifacts in the segmented character's image

Figure 12: Connected components with overlapping bounding boxes and noise

character also contains some noise, which the character segmentation algorithm is able to remove from the final segmented image of the character.

3.5 Video Timestamping

The ability to establish correspondence between the handwritten and the spoken content is essential for enabling audio-video based handwritten mathematical content recognition from classroom videos. An important factor that helps establish this correspondence is the proximity of the occurrence of handwritten and spoken content in the recorded video, which requires us to determine the timestamps associated with the handwritten and the spoken content contained in the video. For a given handwritten character from an input classroom video, we define video timestamp as the time when the character is first fully visible in the video. An audio timestamp, for a given handwritten character, is defined as the time that corresponds to the utterance of the character in the audio content of the classroom video. Our recognition system has the capability to automatically determine these timestamps and these are referred to as the automatic video timestamp TS_V^a and the automatic audio timestamp TS_A^a. For the purpose of evaluating the accuracy of the automatically determined timestamps, we also manually label the training and the test data sets

Algorithm 4: Simple Video Timestamping

input : Segmented Characters `inputSegChars`, Duration of Interest `DOI`, Classroom Video `V`, Sampling $\texttt{N}_{\texttt{sample}}$
output: Video Timestamps `timestampsInputSegChars`

```
1 for i ← startFrame(DOI) to endFrame(DOI) increment-by N_sample do
2     currentSegChars ← characterSegmentation(frameAtIndex(V, i));
3     for j ← 1 to count(currentSegChars) do
4         for k ← 1 to count(inputSegChars) do
5             if timestampsInputSegChars[k] = nil then
6                 if CC_Match(currentSegChars[j], inputSegChars[k]) = true
                  then
7                     timestampsInputSegChars[k] = frameTime(i);
```

and the corresponding timestamps are referred to as the manual video timestamp TS_V^m and the manual audio timestamp TS_A^m. It is necessary to remember that the video timestamp is meant to correspond to the time instant when the character is first fully visible in the classroom video and not the time when the instructor writes the character on the whiteboard. These two time instants may be significantly different when there is occlusion caused by the instructor.

The remainder of this section describes two algorithms for determining video timestamps. The simple video timestamping algorithm makes use of a single binarization threshold for the entire video frame and in many cases the automatic video timestamp is impacted by the presence of the instructor's shadow. The adaptive video timestamping algorithm makes use of a different binarization threshold for regions of the video frame that may have been impacted by the instructor's shadow and is therefore able to determine automatic video timestamps that are more accurate.

3.5.1 Algorithm - Simple Video Timestamping

For a given set of segmented characters obtained from a FOI, the timestamping algorithm iterates over every $\texttt{N}_{\texttt{sample}}$th frame in the corresponding DOI and attempts

to determine if that frame contains the first occurrence of any of the segmented characters and if it does, those characters are assigned a video timestamp corresponding to the time of the current frame. A concise listing of the algorithm in pseudo-code format is included as Algorithm 4 where we consider the set of segmented characters from a FOI to be the input.

1. ***Connected component analysis for every*** $\texttt{N}_{\texttt{sample}}$***th frame:*** In the duration of interest corresponding to the frame of interest from which the *input segmented characters* have been extracted, for every $\texttt{N}_{\texttt{sample}}$th frame, the character segmentation procedure is used to determine the list of segmented characters contained in the frame, which will be referred to as the *current segmented characters* in this algorithm. If the intermediate outputs of the thresholding and connected component analysis for each of the video frames in the earlier preprocessing steps are stored, then this step would not be necessary. As mentioned above, we chose to not store the intermediate results and instead repeat some of the preprocessing steps as the size of such intermediate results can be prohibitively large even for moderate sized videos.

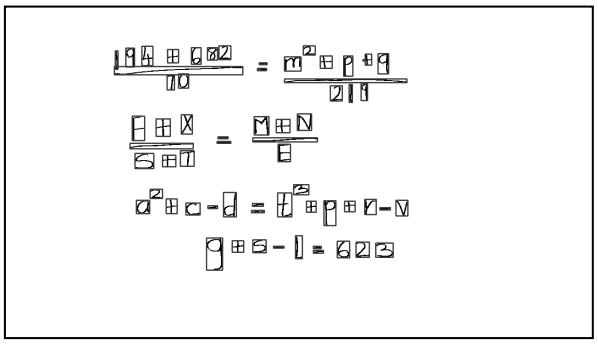
2. ***Finding 'similar' connected components and assigning timestamps:*** For every input segmented character that has not been timestamped, in order to detect whether any of the current segmented characters (received from the previous step) are 'similar' to it , we make use of a similarity check function that is based on a set of features computed for the input segmented character and every current segmented character. This similarity check function, called `CC_Match`, is explained below. If the non-timestamped input segmented character and a current segmented character are deemed to be 'similar' by the `CC_Match` function, then the time of the current frame is considered to be the time of first occurrence of the input segmented character in the classroom video and it is output

as its automatic video timestamp. That particular input segmented character is considered to be timestamped at this stage and is therefore not considered for further iterations of the timestamping process with the remaining frames of the video.

3.5.2 `CC_Match` Function

The detection of similarity between a non-timestamped input segmented character and a current segmented character found in the current frame, in our video timestamping algorithm, relies on a set of three features ($\mathtt{f}_1, \mathtt{f}_2, \mathtt{f}_3$). These features include the Euclidean distance between the center of the bounding-boxes ($\mathtt{f}_1$), the ratio (small number over the larger number) of the area of the bounding-boxes ($\mathtt{f}_2$) and the ratio (smaller number over the larger number) of the number of black pixels in the connected components ($\mathtt{f}_3$). In our implementation of the video timestamping algorithm, a connected component and a segmented character are considered to be similar if each of the conditions ($\mathtt{f}_1 \leq \tau_{\mathtt{f1}}$) , ($\mathtt{f}_2 \geq \tau_{\mathtt{f2}}$) and ($\mathtt{f}_3 \geq \tau_{\mathtt{f3}}$) are satisfied. The specific values for the parameters $\tau_{\mathtt{f1}}$,$\tau_{\mathtt{f2}}$ and $\tau_{\mathtt{f3}}$ used in our implementation are listed in Table 3.

3.5.3 Example - Simple Video Timestamping



(a) Output of connected component analysis for frame at t=140.8 sec (FOI)

Image	Location Coordinates	Video Timestamp (sec)
	(241, 96, 245,134)	9.4
	(264, 90, 281, 129)	10.8
	(300, 86, 321, 126)	12.0
	(355, 93, 377, 120)	13.4
	(407, 89, 426, 123)	14.4
	(442, 89, 462, 114)	16.2
	(467, 85, 491, 113)	16.6

(b) Sample segmented characters with corresponding video timestamps

Figure 13: Video timestamps generated for sample segmented characters

For a given FOI and the corresponding segmented characters (shown in Figure 13(a)), the video timestamps determined by our implementation of the video timestamping algorithm are shown in Figure 13(b). The video timestamp along with the segmented character's image and location coordinates which are obtained from the character segmentation output, as mentioned in Section 2.2, forms the segmented character's feature vector $\mathbf{s} = [s, s_t, s_l]$. The automatic video timestamps shown in Figure 13(b) were found to be very close to the manual video timestamp values. However, it is observed that this is mostly true for characters written in the upper half of the whiteboard which are less affected by shadows. Several characters written in the lower half of the whiteboard were observed to have been affected by shadows caused by the instructor and this resulted in the delayed detection of the automatic video timestamp relative to the corresponding manual video timestamp (*i.e.* when first visible). This is observed in Figure 14 which shows the absolute difference between the automatic TS_V^a and manual TS_V^m video timestamps for each of the character (in the order in which they are visible). Although for the majority of the characters, the absolute timestamping difference was observed to be less than 1 second, for some characters in the lower half of the whiteboard, this difference was found to be much higher (here, a maximum value of 9 seconds). An improved video timestamping technique, that addresses the problem of shadows, is presented in Section 5.3.1.

3.5.4 Algorithm - Adaptive Video Timestamping

The procedure for determining the video timestamps using the *adaptive video timestamping* approach is very similar to the one described above, except for the special handling of handwritten characters whose first appearance on the whiteboard is impacted by the instructor's shadow. As a result of the shadow, the region of the whiteboard in the vicinity of the impacted handwritten character may be significantly

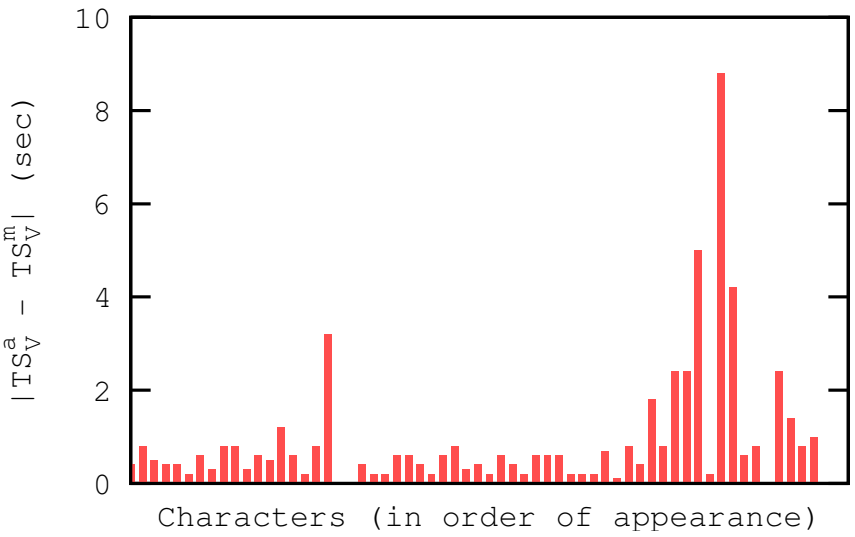


Figure 14: Absolute difference between the automatic (TS_V^a) and manual (TS_V^m)video timestamps for each segmented character of the classroom video

darker than other regions of the whiteboard and in several cases the chosen binarization threshold ($\tau_{\texttt{charBinarize}}$) may cause such regions to be categorized as black. This renders the impacted character virtually invisible to our segmentation procedure as the character (black) merges with the black whiteboard region in the vicinity. To overcome this problem, the adaptive video timestamping approach, for each frame, determines if the character to be timestamped is being impacted by the instructor's shadow and uses a different binarization threshold $\tau_{\texttt{shadow}}$ for locating the occurrence of such characters.

The pseudo-code representation for the adaptive video timestamping algorithm is shown as Algorithm 5. Compared to the simple video timestamping algorithm, we determine two sets of segmented characters for every $\texttt{N}_{\texttt{sample}}$th video frame that is used for determining the timestamps. While one set of segmented characters is determined using the threshold $\tau_{\texttt{charBinarize}}$, the other set is determined using the threshold $\tau_{\texttt{shadow}}$. Next, similar to the earlier video timestamping algorithm, the algorithm proceeds to locate the occurrence of any remaining non-timestamped characters in the set of segmented characters from the current video frame. However, there are two sets

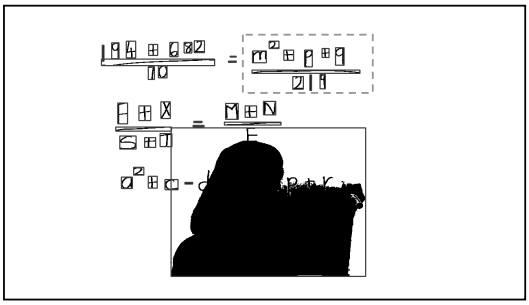
Algorithm 5: Adaptive Video Timestamping

input : Segmented Characters `inSegChars`, Duration of Interest `DOI`, Classroom Video `V`, Sampling $\mathtt{N_{sample}}$, Thresholds $\tau_{\mathtt{charBinarize}}$, $\tau_{\mathtt{shadow}}$

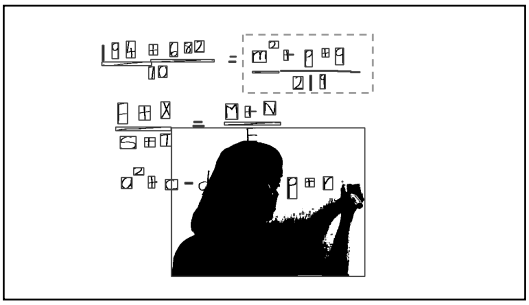
output: Video Timestamps `timestampsInSegChars`

```
1  for i ← startFrame(DOI) to endFrame(DOI) increment-by N_sample do
2      currFrame ← frameAtIndex(V, i);
3      currSegChars ← characterSegmentation(currFrame, τ_charBinarize);
4      shadowSegChars ← characterSegmentation(currFrame, τ_shadow);
5      for j ← 1 to count(inSegChars) do
6          if timestampsInSegChars[k] = nil then
7              biggestBBox ← boundingBox(findBiggestBlob(currSegChars));
8              inCharBBox ← boundingBox(inSegChars[j]);
9              if intersect(biggestBBox, inCharBBox) = true then
10                 for k ← 1 to count(currSegChars) do
11                     if CC_Match(currSegChars[k], inSegChars[j]) = true then
12                         timestampsInSegChars[j] = frameTime(i);
13             else
14                 for k ← 1 to count(shadowSegChars) do
15                     if CC_Match(shadowSegChars[k], inSegChars[j]) = true then
16                         timestampsInSegChars[j] = frameTime(i);
```

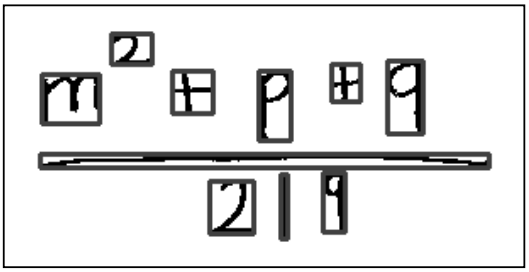
of segmented characters for performing the lookup, and the choice is based on the determination if the character being located is being impacted by the instructor's shadow or not. If the character being timestamped is determined as being impacted by the shadow then we make use of the set of segmented characters that is based on the threshold $\tau_{\mathtt{shadow}}$, while the set based on the threshold $\tau_{\mathtt{charBinarize}}$ is used for other characters. To determine if a character is impacted by the shadow we make use of the bounding box around the instructor (*i.e.* the biggest bounding box). If any portion of the character being timestamped falls within this bounding box, then we designate the character as being impacted by the shadow. The remainder of the algorithm is again similar to the procedure for video timestamping described earlier.

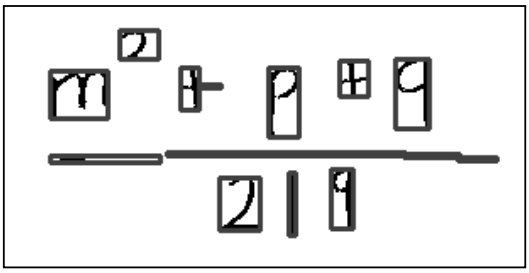


(a) Frame binarized using threshold $\tau_{\text{charBinarize}}$

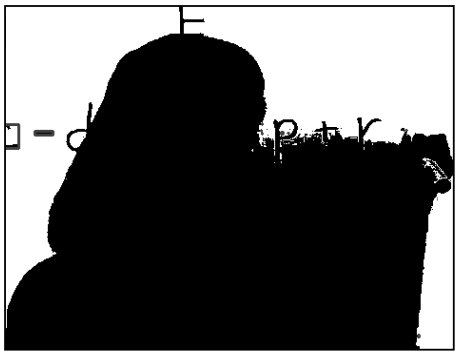


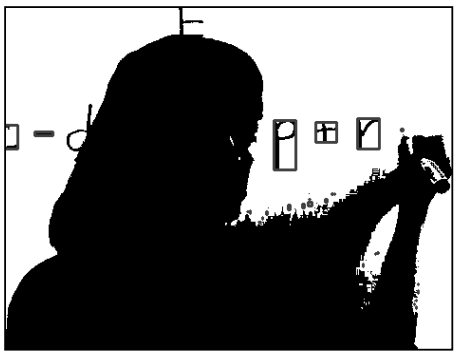
(b) Frame binarized using threshold τ_{shadow}





(c) Using τ_{shadow} on parts of frame without a shadow results in incorrect segmentation – notice the '*divide-by*' line and '+' signs





(d) Using τ_{shadow} on parts of frame with a shadow results in better segmentation – notice the characters '*p*', '+' and 'r'

Figure 15: Binarization with different thresholds for improving timestamping accuracy

3.5.5 Example - Adaptive Video Timestamping

The binarization threshold $\tau_{\texttt{shadow}}$ is lower than the usual character binarization threshold $\tau_{\texttt{charBinarize}}$ and is such that the impacted region of the whiteboard that was previously being categorized as black starts being categorized as white. An obvious question at this stage is why the threshold $\tau_{\texttt{shadow}}$ cannot be used for the entire whiteboard. This is because the threshold $\tau_{\texttt{charBinarize}}$ is determined based on the assumption that there are no shadows. If we use the threshold $\tau_{\texttt{shadow}}$ for regions

without the shadow then there is a possibility that it may lead to significant thinning of the strokes in the handwritten characters and may even lead to cases where a single connected component may get segmented into multiple smaller connected components. The impact of different binarization thresholds on regions of a whiteboard with and without the shadows is shown in Figure 15. Figure 15(c) shows that using $\tau_{\texttt{shadow}}$ for regions of the whiteboard without any shadows leads to incorrect segmentation, while it leads to correct segmentation in parts of the whiteboard with shadows, *i.e.* the region around the instructor and this is shown in Figure 15(d).

3.6 Parameter Values and Segmentation Accuracy

This section contains the values of the parameters used in our implementation of the preprocessing stage of our end-to-end system along with a discussion about the values chosen. We also report the segmentation accuracy achieved by our preprocessing stage.

3.6.1 Parameter Values

The parameters used by the various components of the preprocessing stage have been listed in Table 3 along with a brief description as well as the values that were assigned in our implementation of the end-to-end system. Most of these parameters have been empirically estimated over a small representative set of input videos or have been manually determined by closely inspecting the input videos as well as the output of the intermediate stages of preprocessing. Manual determination of the parameters were based on the average value and the range of values for measures such as the size of the bounding box of a character, the number of black pixels in a character, the distance between two adjacent characters, the thickness of the stroke of a character, the thickness of the whiteboard and the whiteboard jitter caused due to writing on the whiteboard.

Table 3: Values of the parameters used in the various components of preprocessing

Symbol	Description of the Parameter	Value
$\tau_{\texttt{ROIBinarize}}$	Binarization threshold used for detecting the ROI	102
$\texttt{d}$	Number of pixels by which we dilate the ROI in each direction	4
$\texttt{e}$	Number of pixels by which we erode the ROI in each direction	10
$\texttt{N}_{\texttt{sample}}$	Frame sub-sampling factor *i.e.* every $\texttt{N}^{\texttt{th}}_{\texttt{sample}}$ frame will be selected	6
$\tau_{\texttt{charBinarize}}$	Binarization threshold used for detecting characters/connected components	35
$\tau_{\texttt{minPixels}}$	Pruning threshold - minimum number of pixels in a connected component for it to be considered a character	20
$\tau_{\texttt{maxPixels}}$	Pruning threshold - maximum number of pixels in a connected component for it to be considered a character	1500
$\tau_{\texttt{DOIDetect}}$	Threshold used to detect the initial DOIs based on whether the number of connected components in the component count plot transition from a value above $\tau_{\texttt{DOIDetect}}$ to a value below or equal to it	5
$\tau_{\texttt{FOIMatch}}$	Threshold used in the `FOI_Match` function to compare the ratio of the number of pairs of 'similar' connected components between the two input FOIs to the minimum of the number of connected components in each of the two input FOIs	0.75
$\tau_{\texttt{shadow}}$	Binarization threshold used for detecting characters/connected components in regions impacted by instructor's shadow	25
$\tau_{\texttt{f1}}$	Threshold used for the feature f_1 in the `CC_Match` function	9
$\tau_{\texttt{f2}}$	Threshold used for the feature f_2 in the `CC_Match` function	0.75
$\tau_{\texttt{f3}}$	Threshold used for the feature f_3 in the `CC_Match` function	0.75

Parameters $\tau_{\mathbf{ROIBinarize}}$ and $\tau_{\mathbf{charBinarize}}$: The reason why the binarization thresholds $\tau_{\texttt{ROIBinarize}}$ and $\tau_{\texttt{charBinarize}}$ take significantly different values of `102` and `35` although both are used to binarize a grayscale image to a black and white image is that $\tau_{\texttt{ROIBinarize}}$ operates on the grayscale image of a video frame (the calibration frame) while $\tau_{\texttt{charBinarize}}$ operates on the frame differencing output image which tends to have very low intensity values due to the differencing operation.

Parameters d and e : We chose to dilate the initial ROI by a `d = 10` pixels which is based on the minimum distance of the white region of the whiteboard to any edge of the video frame and the thickness of character strokes (mostly `3-8` pixels). This value for `d` will be able to remove any character strokes left behind on the calibration frame by poor erasure and not dilate beyond the frame of the whiteboard which separates the white connected component of the whiteboard from merging with that of the background wall. After this, we erode the dilated ROI by `e = 16` pixels and therefore the final ROI is `(e-d) = 6` pixels smaller (on all sides) than the initial ROI thereby being able to remove artifacts caused by jitter (less than `6` pixels) of the whiteboard when writing on it. Note that the values of the parameters `d` and `e` are set in terms of pixels and these values may need to be adjusted if we use a camera with a significantly different resolution.

Parameter $\mathbf{N_{sample}}$: The input video has `30` frames per second and a sampling factor of `6` implies that we chose about `5` frames per second for preprocessing. The only reason for sub-sampling the frames is to speed up the execution of our preprocessing algorithm. One must note that while sub-sampling result in timestamps that are accurate to `0.20` seconds only, such accuracy is sufficient for the proposed recognition system where these timestamps are used to assist in establishing correspondence between the handwritten and the spoken content.

Parameters $\tau_{\mathbf{minPixels}}$ and $\tau_{\mathbf{maxPixels}}$: The values of $\tau_{\texttt{minPixels}}$ and $\tau_{\texttt{maxPixels}}$ are determined by studying the histogram of the number of black pixels in each segmented

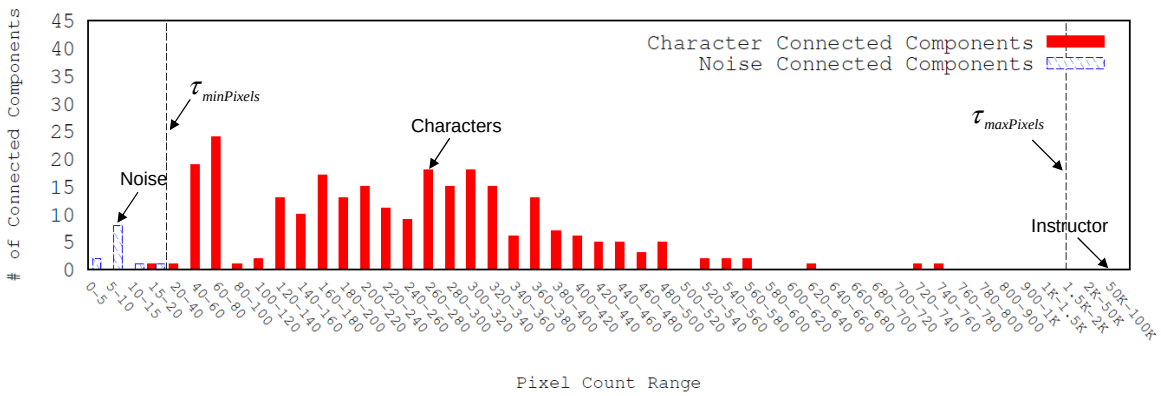


Figure 16: Distribution of character and noise connected components by size for 4 sample FOI frames. The dotted lines represent the thresholds $\tau_{minPixels}$ and $\tau_{maxPixels}$. The total number of connected components is 273.

character compared to those of small noise connected components and very large connected components such as the instructor, his/her shadow and the frame of the whiteboard. Figure 16 uses a histogram to show the distribution of connected components, both noise and characters, over different pixel ranges. The distribution shown in the histogram is for 4 video frames (that are FOIs) selected from 4 different video segments and the total number of connected components in the histogram is 273. Towards the left end of the distribution we chose $\tau_{\texttt{minPixels}} = 20$, to separate the connected components that correspond to characters and the connected components with very small number of pixels that correspond to noise. There may be some overlap between the histogram values for the character connected components and the small noise connected components but the error caused by this overlap is fairly low (about 1.1%, as reported in Section 3.6.2). Since the histogram is plotted for frames that have been passed through the ROI filter, the connected component corresponding to the whiteboard frame or any noise outside the whiteboard frame are not included here. Also, we use FOI frames here, which, by definition, do not have the instructor and his/her shadow overlapping with any part of the ROI and therefore the corresponding large connected components are not shown here. We found, by observation,

that when the instructor is fully within the frame, the corresponding connected component tends to have about 50K black pixels. The characters usually had less than 500 black pixels and the long horizontal lines representing algebraic division had less than 800 black pixels. Therefore, the second pruning threshold of $\tau_{\texttt{minPixels}} = 1500$ was sufficient to detect when a connected component with more than $\tau_{\texttt{maxPixels}}$ black pixels (such as the instructor and his/her shadow) overlapped with the ROI and this was used in the occlusion plot.

Parameter $\tau_{\texttt{DOIDetect}}$: A value of 5 has been assigned to $\tau_{\texttt{DOIDetect}}$ as opposed to a value of 0 to mark the end of the DOI to allow for some connected components caused by noise. However, we have observed that due to the reasonably good illumination conditions, the ROI filtering, the pruning of connected components to remove noise and the requirements for the FOI detection, there are very few such noise artifacts remaining in the FOIs and therefore a value of 0 may have also worked for $\tau_{\texttt{DOIDetect}}$.

Parameter $\tau_{\texttt{FOIMatch}}$: A value of 0.75 has been assigned to $\tau_{\texttt{FOIMatch}}$ *i.e.* a match of at least 75% between the connected components of the two FOIs for the corresponding DOIs to be merged. Given the good performance of most of our preprocessing components and the controlled recording environment, a higher value for this parameter may also have worked. An unintented but desirable consequence of using this value for the parameter $\tau_{\texttt{FOIMatch}}$ is that the system can handle erasures of about 25% of the characters as long as the erasures are before the detected FOI.

Parameter $\tau_{\texttt{shadow}}$: The threshold $\tau_{\texttt{shadow}}$ is slightly lower (25) than the threshold value $\tau_{\texttt{charBinarize}}$ to accommodate for the shadows and classify more pixels as white.

Parameters $\tau_{\texttt{f1}}$, $\tau_{\texttt{f2}}$ and $\tau_{\texttt{f3}}$: The value of $\tau_{\texttt{f1}}$ is set to a value of 9 since we have estimated that the maximum whiteboard jitter tends to be less than 6 pixels, so a shift of 6 pixels in the horizontal and/or vertical axis would lead to a shift in the center of the bounding box which is less than $\sqrt{(6^2 + 6^2)} = 8.49 \approx 9$ pixels. In this case, since we compare two FOIs where the instructor is not touching the whiteboard, the

shift tends to be much lesser. The values of $\tau_{\texttt{f2}}$ and $\tau_{\texttt{f3}}$ which correspond to the ratio of the area of the bounding boxes and the ratio of the number of black pixels in the connected components are both set to `0.75` *i.e.* a match of at least `75%`. Again, due to the good performance of our preprocessing algorithms and the controlled recording environment used for our dataset, higher values may be used for these parameters without leading to any significant changes in the preprocessing results.

3.6.2 Segmentation Accuracy

The aim of video preprocessing stage is to generate a set of segmented characters along with the corresponding location coordinates and the timestamp. In this section, we report the accuracy of character segmentation for the test data set `DS-TEST-1` in terms of the following values (1) *character segmentation accuracy* - the percentage of actual characters that appeared in the set of segmented characters is `99.3`% (2) *false negative rate* - the percentage of actual characters that failed to appear in the set of segmented characters `0.7`% and (3) *false positive rate* - the percentage of segmented characters that do not correspond to actual characters is `1.1`%. It is evident that the video preprocessing stage is able to achieve a significantly high accuracy for character segmentation, part of which can also be attributed to the conformance of the recorded video to the assumptions that were stated at the beginning of this chapter.

3.7 Summary

In this chapter we presented a set of video preprocessing techniques that can be used for generating a set of segmented characters from the input classroom videos and the associated meta-information for the subsequent stages. We outlined a set of assumptions that need to be conformed to by the input classroom video for our techniques to perform optimally. We presented a technique for determining the region of interest, which is roughly the area of the whiteboard inside of the whiteboard frame. A set of techniques were presented for determining the durations of interest

and the corresponding frames of interest. The duration of interest, in our system, corresponds to the time between two successive complete erasures of the whiteboard and the frame of interest, in a given duration of interest, is a frame that contains all the characters handwritten in that duration and no occlusion of the region of interest. To retrieve the handwritten characters from a given frame of interest and to generate the corresponding timestamp information, we proposed techniques for character segmentation and video timestamping. Finally, we provide the values for the various parameters used by the video preprocessing stage and present results for the character segmentation accuracy.

CHAPTER IV

AMBIGUITY DETECTION & OPTION SELECTION

4.1 Introduction

The main premise behind combining the output of multiple recognizers is to design an approach that helps improve the recognition accuracy, compared to that of a single recognizer. However, combination, as it turns out, may not always lead to an improvement in the recognition accuracy and, in many cases, may result in an incorrect output even when one or more individual recognizers generate the correct output, leading to a deterioration in the recognition accuracy. In the context of the audio-video based recognition system described in this dissertation, the audio text recognizer generates audio options that corroborate or contradict the video options generated by the video text recognizer for each segmented character. It is possible that the audio options generated by the audio text recognizer upon combination with the video options will introduce more errors than corrections in the video recognizer's output and, as a result, deteriorate the end-to-end character recognition accuracy. This chapter focuses on *ambiguity detection* and *option selection* techniques that can be used in preparation for the combination stage to reduce the errors introduced during combination and to improve the chances of correcting the errors in the video text recognizer output.

The need for the ambiguity detection and option selection components, shown in Figure 17, is motivated by two key observations [111]. First, in a two-stage recognition system, where the video text recognizer acts as the primary recognizer, the output of the primary recognizer is correct for a significant fraction of the input. While forwarding the correctly recognized segmented characters for combination with the

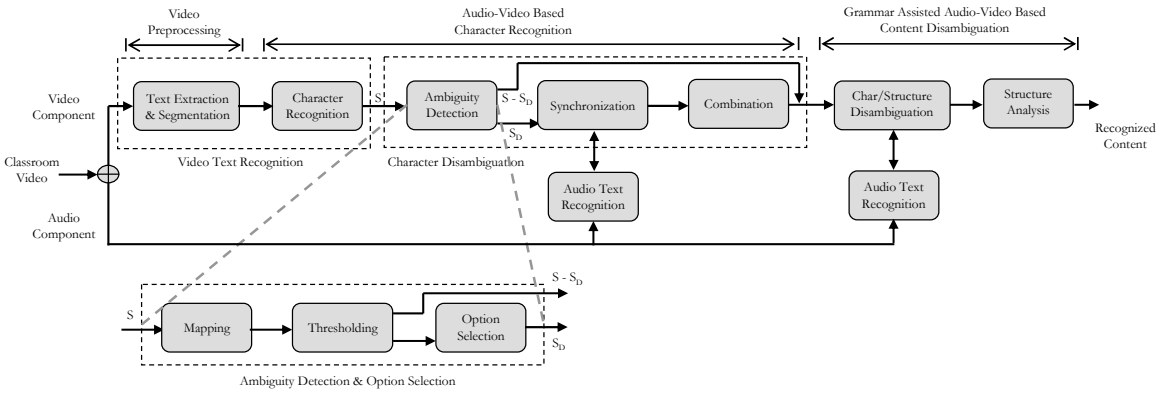


Figure 17: Ambiguity detection & option selection

audio text recognizer opens up the possibility of introducing errors in the output of the primary recognizer, not forwarding the incorrectly recognized segmented characters for combination eliminates any chance of correcting the corresponding recognition output for such characters. This serves as the motivation for ambiguity detection, which attempts to determine the subset of segmented characters that are more likely to have been recognized incorrectly by the video text recognizer and forwards only this subset for subsequent combination with the output of the audio text recognizer. Second, in our implementation of the audio-video based recognition system, the video text recognizer produces multiple video options for each character, which may be used for combination with the output from the audio text recognizer. Selecting too many or too few video options may increase the chance of errors as opposed to corrections in the output of the audio-video based combination component. This observation serves as the motivation for the option selection step, which involves selecting the appropriate subset of video options to be forwarded for combination. These two observations outlined above form the basis of the techniques proposed in this chapter. Specifically, we make the following contributions:

- *Ambiguity Detection* - We model ambiguity detection as a two-step process consisting of mapping and thresholding, techniques for which are described and evaluated. Our evaluation demonstrates a significant improvement in the

end-to-end character recognition accuracy that can be achieved by using the proposed techniques for mapping and thresholding.

- *Option Selection* - Option selection techniques that select a fixed or variable number of options from the original set of video options have been proposed and experimentally evaluated. Again, our evaluation demonstrates a significant improvement in the end-to-end character recognition accuracy that can be achieved by making use of option selection.

The remainder of this chapter is organized as follows. Section 4.2 describes the various ambiguity detection techniques that we have implemented for the end-to-end recognition system and Section 4.3 describes the option selection techniques. An experimental evaluation is presented in Section 4.4, which is followed by a summary of the findings that are presented in Section 4.5.

4.2 Ambiguity Detection

In the proposed audio-video based recognition system the output of the primary, video text recognizer, consists of both - the characters that have been recognized correctly and the characters that have been recognized incorrectly. While the accuracy for the subset of characters that have been recognized correctly by the video text recognizer (which is 100% to begin with) can only decrease or remain constant as a result of combination; the accuracy, after combination, for the subset of characters that have been recognized incorrectly by the video text recognizer (0% to begin with) can only increase or remain constant. While, in most scenarios, it is not possible to accurately identify the two subsets of characters, it is often possible (using match scores generated by the video text recognizer and/or training data sets) to identify the characters that have a higher likelihood of being correctly or incorrectly recognized by the video text recognizer. Ambiguity detection techniques, proposed in this chapter, attempt to identify the subset of characters that have a higher likelihood of being

recognized incorrectly by the video text recognizer (termed the set of *ambiguous* characters) and trigger combination only for this subset, while forwarding the other subset (termed the set of *non-ambiguous* characters) unmodified to the subsequent components of the recognition system.

The proposed ambiguity detection techniques, however, are faced with an interesting dilemma. If the technique imposes very stringent tests for establishing the correctness of the video option then many correct video options will get categorized as ambiguous and are therefore exposed to the possibility of being rendered incorrect as a result of combination. On the other hand, if the technique does not impose stringent tests for establishing correctness then some of the incorrect video options will get categorized as non-ambiguous and will therefore not have the possibility of being corrected via combination with audio options. Our aim, with the help of ambiguity detection, is to determine, as accurately as possible, the two complementary subsets of incorrectly and correctly recognized segmented characters. We model ambiguity detection as a function D that is designed to operate on the set of segmented characters S and returns S_D, the set of ambiguous characters. The set of *non-ambiguous* characters is, therefore, $S - S_D$. If the ambiguity detection function D is well designed, we may see an end-to-end character recognition accuracy that is higher than the recognition accuracy achieved by passing the entire test set through just the video text recognizer or passing the complete set through the audio-video based combination component (*i.e.* without ambiguity detection). Ambiguity detection can be divided into two main tasks of mapping and thresholding, which are described below.

4.2.1 Mapping

Video match scores that are generated by the video text recognizer are often not the best basis for ambiguity detection and classifier combination. This happens when the generated video match scores are not normalized or when the test data set is

significantly different from the one for which the recognizer has been trained. We may map (*i.e.* rescore) these values to a new set of values, and if done correctly, a good mapping of the video match scores can result in improving the accuracy of the generated output, help in better detection of ambiguous characters, better selection of options to forward to the audio-video based combination components and possibly better combination with the audio options generated by the audio text recognizer. Some commonly used score normalization techniques have been discussed in [48].

Match score to conditional probability mapping. The mapping technique that we have used to recompute the video match score for each character s is to calculate the value of the conditional probability that the character is correctly classified, *i.e.* $v_i^c(s) = G(s)$, given that the video match score is equal to $v_i^p(s)$. We represent the new video match score for the character s using $v_i^{p\prime}(s)$,

$$v_i^{p\prime}(s) = Prob\{v_i^c(s) = G(s) | v_i^p(s)\} = \frac{Prob\{v_i^c(s) = G(s), v_i^p(s)\}}{Prob\{v_i^p(s)\}} \tag{13}$$

Estimating these conditional probabilities is done using a training set S_{train} that consists of a set of segmented characters, s, along with their ground truth, $G(s)$. This conditional probability, which will be used as the new video match score, can be estimated as follows,

$$v_i^{p\prime}(s) = \frac{\text{Number of times } v_i^c(s) \text{ is correctly classified with a score } v_i^p(s)}{\text{Number of times } v_i^c(s) \text{ has a score } v_i^p(s)} \tag{14}$$

After mapping, the video match scores for each video option change, and need to be reordered so that they are in decreasing order. Figure 18 shows a set of three video options for seven different characters before and after mapping. The options that are highlighted in gray are the ones that were reordered as a result of the new video scores. Although mapping is optional, as we will see in Section 4.4, the use of a suitable mapping technique may lead to significant improvements in the recognition

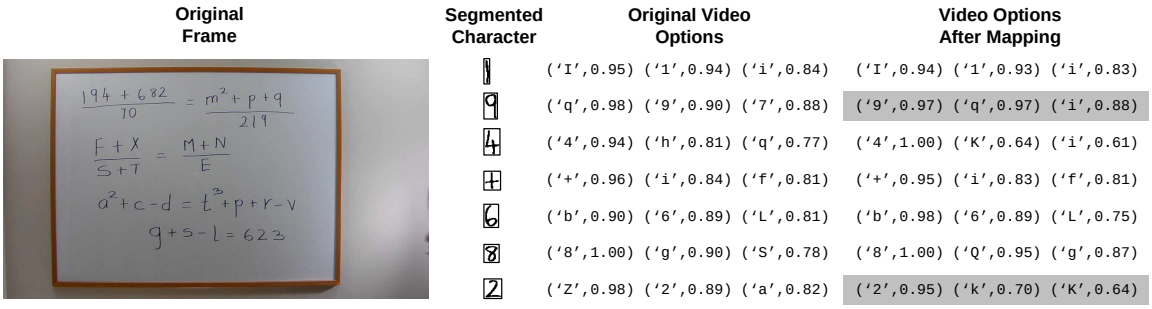


Figure 18: Showing video options before and after mapping for a few segmented characters

accuracy. In some cases, such as in the case of a limited training set, it may be advisable to divide the entire range of match scores into sub-ranges and then compute the value of the conditional probability given that $v_i^p(s)$ falls within a certain range of values instead of a specific value.

4.2.2 Thresholding

We now present a set of thresholding conditions that, based on the output of the video text recognizer, could be used to determine if a segmented character $s \in S$ should be considered ambiguous. These conditions can be divided into two broad categories: simple thresholding and character-specific thresholding.

4.2.2.1 Simple Thresholding

Simple thresholding techniques are used to determine if a segmented character element $s \in S$ is ambiguous or not based on one or more thresholding conditions that are uniformly applied to all the characters. Our first ambiguity detection method considers the segmented character element $s \in S$ to be ambiguous if the video match score corresponding to the first video option $v_1^p(s)$ (which is the highest amongst all the video match scores) is less than a predetermined absolute threshold value $AbsThr$. Mathematically, the above ambiguity detection technique can be represented as follows

$$D(S) = \{s \in S \mid v_1^p(s) < AbsThr\} \tag{15}$$

Another method for ambiguity detection, used in our implementation, compares the ratio between the match scores of first and second video options to a predetermined relative threshold value $RelThr$. The ambiguity detection technique, in this case, can be mathematically expressed as follows

$$D(S) = \{s \in S \mid v_2^p(s)/v_1^p(s) > RelThr\} \tag{16}$$

The method is based on the premise that the top video option, if correct, will have its match score well separated from the match score for the second best video option. Also note that each of the methods presented above make use of the same threshold for all the characters.

4.2.2.2 Character-Specific Thresholding

The use of a single threshold for all characters may not be the best thresholding technique. This could be attributed to the fact that the accuracy of a given recognizer is often different for different characters. To exploit this fact and to take into account this characteristic of the recognizers, we intend to determine a different threshold value $T(c)$ for each character $c \in C$. Such character-specific thresholds can then be used for thresholding as follows

$$D(S) = \{s \in S \mid v_1^p(s) < T(v_1^c(s))\} \tag{17}$$

where $T(v_1^c(s))$ corresponds to the character-specific threshold for character represented by $v_1^c(s) \in C$.

To determine the values for the character-specific thresholds $T(c)$, we rely on the test data set. We use S_{train} to represent the set of segmented characters contained in

the test data set. We start by partitioning the set S_{train} into multiple disjoint subsets $S(c)$, each corresponding to a character $c \in C$,

$$S(c) = \{s \in S_{train} \mid v_1^c(s) = c\} \tag{18}$$

Our goal is to determine a character-specific threshold T that maximizes the audio-video recognition accuracy $\alpha(S(c), T)$ for each such subset $S(c)$ and as a consequence, maximizes the accuracy for the set S_{train}. The threshold value T, thus determined for the subset $S(c)$, is the character-specific threshold $T(c)$ and is given by

$$T(c) = \arg\max_T(\alpha(S(c), T)) \tag{19}$$

To determine $\alpha(S(c), T)$, we proceed as follows. Each character in $S(c)$ is categorized into one of the three sets: $S_1(c)$, $S_N(c)$ and $S_\infty(c)$. Here, $S_1(c)$ refers to those elements of $S(c)$ whose first video option is correct,

$$S_1(c) = \{s \in S(c) \mid v_1^c(s) = G(s)\} \tag{20}$$

$S_N(c)$ refers to those that have the correct video option within the first N video options but not the first video option,

$$S_N(c) = \{s \in S(c) \mid \exists x : v_x^c(s) = G(s) \quad \wedge \quad 1 < x \leq N\} \tag{21}$$

and $S_\infty(c)$, given by

$$S_\infty(c) = \{s \in S(c) \mid \exists x : v_x^c(s) = G(s) \quad \wedge \quad x > N\} \tag{22}$$

refers to those that do not have the correct video option in the first N options and therefore cannot be corrected by our system as the correct video options will not be available to the audio-video based combination component. Note that, for mathematical simplicity, we are assuming that only the top N options are forwarded for

audio-video based combination.

To maximize the accuracy $\alpha(S(c), T)$, we focus on the sets $S_1(c)$ and $S_N(c)$. For the characters in $S_1(c)$, which is the set of characters recognized correctly by the video text recognizer, we do not wish to introduce errors by forwarding such characters to the audio-video based combination component. For the characters in $S_N(c)$, our aim would be to forward as many elements from $S_N(c)$ as possible to the audio-video based combination component by tagging them as ambiguous, since audio-video combination can only result in improving the recognition accuracy for the elements from $S_N(c)$. Since the decision whether or not to forward the character depends on the threshold T, we define four disjoint subsets of $S(c)$ as follows. The first is the set of *True Positives*, $TP(c)$, that contain those characters that are correctly recognized by the video text recognizer and are tagged as non-ambiguous and are, therefore, not passed for audio-video based combination. This set, which has a recognition accuracy value that is equal to one (100%), is given by

$$TP(c) = \{s \in S_1(c) \mid v_1^p(s) \geq T\} \tag{23}$$

The next two subsets are those that contain characters that are forwarded for audio-video based combination. The first is the set of *False Negatives*, $FN(c)$, which contains those characters that are correctly recognized by the video text recognizer but are tagged as ambiguous

$$FN(c) = \{s \in S_1(c) \mid v_1^p(s) < T\} \tag{24}$$

and the second is the set of *True Negatives*, $TN(c)$, which contains those characters that are not correctly recognized by the video text recognizer but are tagged as ambiguous and have the correct character within the first N options,

$$TN(c) = \{s \in S_N(c) \mid v_1^p(s) < T\} \tag{25}$$

Here we will assume that when the audio-video combination component is given a set of N video options with one of the options being correct, the recognition accuracy is equal to $\alpha_F^{N'}$. An approximate value for this accuracy may be estimated from the training data set.

The fourth subset is the set *False Positives*, $FP(c)$, which is the set of all characters that are not correctly recognized, have the correct option within the first N options, and have been tagged as non-ambiguous. This subset, given by

$$FP(c) = \{s \in S_N(c) \mid v_1^p(s) \geq T\} \tag{26}$$

is not passed for audio-video based combination and, therefore will have a recognition accuracy score of zero (0%).

Combining together the four subsets above along with their recognition accuracies, we have the overall recognition accuracy that is given by

$$\begin{aligned} \alpha(S(c), T) &= \frac{1 \times |TP(c)| + \alpha_F^{N'} \times |FN(c)| + 0 \times |FP(c)| + \alpha_F^{N'} \times |TN(c)|}{|TP(c)| + |FN(c)| + |TN(c)| + |FP(c)|} \quad (27) \\ &= \frac{|TP(c)| + \alpha_F^{N'} \times (|FN(c)| + |TN(c)|)}{|TP(c)| + |FN(c)| + |TN(c)| + |FP(c)|} \quad (28) \end{aligned}$$

The value for character-specific thresholds $T(c)$ can now be calculated by substituting the accuracy term in Equation 19 with the formulation shown in Equation 28.

4.3 Option Selection

Another factor that impacts the audio-video character recognition accuracy is the number of video options that are considered for combination for a given segmented character. While it may seem that forwarding all the generated video options for combination increases the chances of having the correct output among them, one should note that in cases when the correct output is present in the top few video options, passing several additional video options to the combination stage may lead

to an increased chance of arriving at an incorrect output. On the other hand, if the number of options passed for combination is too low then one runs the risk of ruling out the correct video option even before combination. If a given segmented character has been designated as being ambiguous, the option selection component, for such ambiguous characters, determines the subset of video options that should be forwarded for audio-video based combination. Option selection aims to increase the chances of having the correct video option in the list of options selected for combination, while simultaneously attempting to decrease the chances of having incorrect options in the list of selected options.

The option selection methods used by our audio-video based recognition system include three different strategies. The first is to simply select the top $NumOpt$ number of video options,

$$O(s) = \{\mathbf{v}_i(s) \in V(s) \mid i \leq NumOpt\} \tag{29}$$

The second is to select those video options that have a video match score that is greater than an absolute threshold $OptAbsThr$,

$$O(s) = \{\mathbf{v}_i(s) \in V(s) \mid v_i^p(s) > OptAbsThr\} \tag{30}$$

The third is to select the video options that have a video match score that exceeds some fraction $OptRelThr$, of the top video match score,

$$O(s) = \{\mathbf{v}_i(s) \in V(s) \mid v_i^p(s)/v_1^p(s) > OptRelThr\} \tag{31}$$

The evaluation of each of these video option selection strategies is presented in Section 4.4.4.

4.4 Experiments

We conducted a set of experiments to evaluate the performance of various mapping, thresholding and option selection techniques proposed in this chapter. First, we experimentally established the baseline accuracy values, both without and with the use of mapping. Then, we conducted end-to-end experiments that examined the effect of the proposed thresholding and option selection techniques on the audio-video based character recognition accuracy. Our results show that the proposed techniques, when effectively utilized, can lead to a significant improvement in the end-to-end character recognition accuracy $\alpha(S)$.

4.4.1 Setup

The effect of various mapping, thresholding and option selection techniques was measured by observing the end-to-end character recognition accuracy, which was determined by comparing the ground truth to the character name contained in the output of the *audio-video based character recognition stage* (as shown in Equations 6, 11 and 12) over the test data set. While the knowledge of the audio-video synchronization and combination techniques is not necessary for understanding the results presented in this section, a placeholder implementation of such techniques was needed for computing the end-to-end character recognition accuracy $\alpha(S)$. Specifically, we made use of a feature rank sum based technique, described in Section 5.4.6, for synchronization and utilized a recognizer-specific weight based technique, described in Section 6.3.2, for combination. To enable comparison between different sets of experiments, all the experiments that did not employ mapping used a value of 0.6 as the video weight w_V and a value of 0.4 as the audio weight w_A. Experiments that employed mapping used values of w_V=0.8 and w_A=0.2. All the experiments described in this chapter made use of the test data set `DS-TEST-1` for evaluation. The training set `DS-TRAIN-1` was used to perform mapping and to estimate the character-specific

thresholds. In the following sections, to effectively highlight any key observations made during an experiment we will list such observations at the beginning of a section, which will be followed by a detailed explanation.

4.4.2 Baseline Accuracies and Mapping

This section establishes the baseline video text recognizer accuracy for the test data set `DS-TEST-1`. The baselines were established for two setups of the end-to-end recognition system, without and with the use of mapping. Experiments were also used to study the effect of mapping on the order of the video options generated by the video text recognizer.

4.4.2.1 Baseline Accuracies & Motivation for Mapping

Key Observations

- *The end-to-end character recognition accuracy deteriorates when all the characters and all the video options are forwarded for combination, thereby motivating the need for ambiguity detection and option selection.*
- *Mapping results in a significant increase in the end-to-end character recognition accuracy when all the characters are passed only through the video text recognizer and also when all the characters with all the video options undergo audio-video combination based recognition.*
- *The end-to-end character recognition accuracy with only the video text recognizer, both with and without mapping, acts as the baseline for future experiments. The end-to-end character recognition accuracy when all the characters and options undergo audio-video combination based recognition, both with and without mapping, also act as a reference for future experiments.*

Explanation. We established the baseline accuracies (with and without mapping) by considering two scenarios. In the first scenario, shown in rows 1 and 2 in Table 4

Table 4: Baselines & motivation for mapping

S.No.	Mapping	Thresholding	Option Selection	Accuracy(%) $\alpha(S)$
Effect of mapping on the VTR accuracy				
1.	No	No (only VTR)	No (n/a)	53.7
2.	Yes	No (only VTR)	No (n/a)	62.2
Effect of mapping on the AVC accuracy				
3.	No	No (all AVC)	No (all options)	50.0[1]
4.	Yes	No (all AVC)	No (all options)	61.4[2]

Setup[1] - Synchronization: `Rank Sum`, Combination: `Weighted (`w_V`=0.6,`w_A`=0.4)`

Setup[2] - Synchronization: `Rank Sum`, Combination: `Weighted (`w_V`=0.8,`w_A`=0.2)`

all segmented characters in S were classified as non-ambiguous. As a consequence the video option with the highest video match score was chosen as the final character recognition output. The recognition accuracy $\alpha(S)$, as a result, is equivalent to the video text recognizer (VTR) accuracy. The baseline VTR accuracy for the test data set `DS-TEST-1` without mapping applied to the video match scores (shown in row 1 of Table 4) was determined to be 53.7%. The VTR accuracy with mapping was determined to be 62.2% (shown in row 2 of Table 4). This value serves as the baseline accuracy value for a system with mapping. The significant increase in character recognition accuracy due to mapping can be attributed to the fact that mapping makes use of the labeled training data set `DS-TRAIN-1` to rescore the video match scores generated by the VTR. Mapping, therefore, is able to incorporate the features of the handwritten characters that are specific to the dataset.

In the second scenario, shown in rows 3 and 4 in Table 4, all the segmented characters in S were classified as ambiguous, and as a result all the characters undergo audio-video based combination (AVC). As opposed to the first scenario, which established the VTR baselines, the second scenario establishes the important reference accuracy values for evaluating the effectiveness of the thresholding and option selection techniques. The character recognition accuracy $\alpha(S)$, in this case was found

to be 50.0% without mapping and 61.4%with mapping. Since the recognition rates, when all the characters with all the video options are forwarded for audio-video based combination, are lower than the corresponding values for the standalone VTR, this motivates the need for thresholding and option selection techniques.

4.4.2.2 Effect of Mapping on Video Option Ordering

Key Observation

- *Given a set of segmented characters, our mapping technique, as compared to the no mapping scenario, improves the chances of having the correct output among the top N video options.*

Explanation. One effect of our mapping technique, as observed in the previous section, is that it helps improve the baseline VTR accuracy by rescoring and, as a result, reordering the video options so that more characters end up with the correct video option as their top video option. In many cases, the top video option, even after mapping is not the correct video option. In such case, having the correct video option among those that are sent to the synchronization and combination stages improves the chances of correct recognition during audio-video based combination. We conducted an experiment to determine the percentage of input segmented characters that had the correct character in the top N options, for different values of N. The results in Figure 19 show that mapping not only improves the baseline VTR accuracy (when N=1), but also helps to improve the chances of having the correct video option among the top N options.

4.4.3 Thresholding Techniques

This section examines the thresholding techniques proposed in this chapter. We start by providing the motivation for thresholding by demonstrating the improvement in A/V character recognition accuracy that can be achieved by making use of

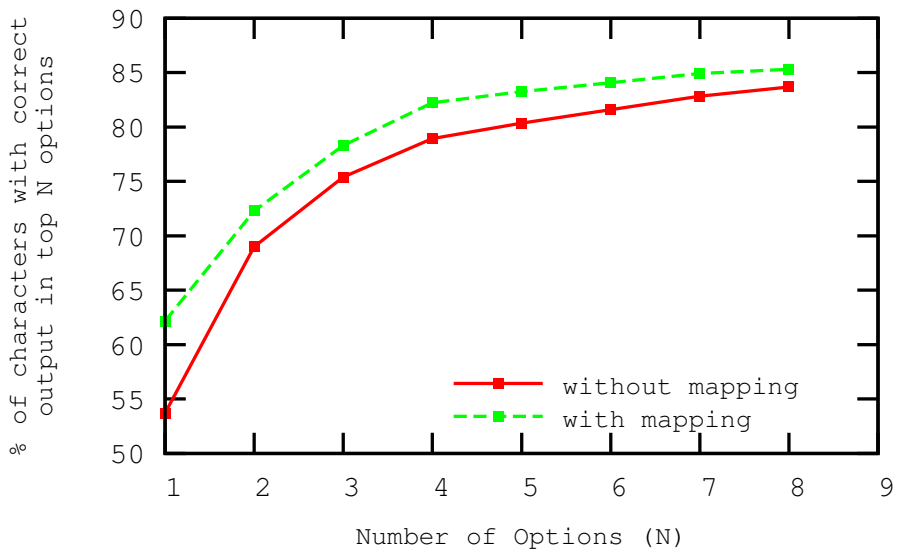


Figure 19: Percentage of segmented characters with the correct output amongst the top N video options

our thresholding techniques, compared to the baselines established earlier. Next, to gain a better understanding of the factors impacting the thresholding component, we examine a specific setup of the absolute thresholding in detail. We also conducted experiments to show the effect of different values of absolute and relative thresholds, and character-specific thresholds on the A/V character recognition accuracy.

4.4.3.1 Motivation for Thresholding

Key Observation

- *The proposed thresholding techniques are effective in improving the A/V character recognition accuracy when compared to the baselines, both without and with mapping, thereby demonstrating the effectiveness of thresholding.*

Explanation. Table 5 examines the effect of thresholding on the recognition accuracy for two different setups, one without mapping and one with mapping. When the option selection component was used, it was configured to forward all video options for audio-video based combination. For each setup, the goal of the experiments was

Table 5: Motivation for thresholding

S.No.	Mapping	Thresholding	Option Selection	Accuracy(%) $\alpha(S)$
Effect of thresholding on the accuracy $\alpha(S)$ without mapping[1]				
1.	No	No (only VTR)	No (n/a)	53.7
2.	No	No (all AVC)	No (all options)	50.0
3.	No	AbsThr=0.90	No (all options)	55.0
4.	No	RelThr=0.95	No (all options)	56.8
5.	No	CharSpec ($\alpha_F^{N'}$=40)	No (all options)	61.8
Effect of thresholding on the accuracy $\alpha(S)$ with mapping[2]				
6.	Yes	No (only VTR)	No (n/a)	62.2
7.	Yes	No (all AVC)	No (all options)	61.4
8.	Yes	AbsThr=0.95	No (all options)	63.8
9.	Yes	RelThr=0.98	No (all options)	64.5

Setup[1] - Synchronization: Rank Sum, Combination: Weighted (w_V=0.6,w_A=0.4)
Setup[2] - Synchronization: Rank Sum, Combination: Weighted (w_V=0.8,w_A=0.2)

to determine if thresholding could overcome the degradation in the recognition accuracy that occurs when all the segmented characters are forwarded for audio-video based combination (rows 2 and 7) and possibly improve the accuracy when compared to the baseline VTR accuracy (rows 1 and 6). As expected, the use of thresholding (rows 3, 4, 5, 8 and 9) results in improving the recognition accuracy when compared to the corresponding baselines. Note that we do not utilize character-specific thresholding with mapping. This is because both the character-specific thresholds and the rescoring done by the mapping component attempt to incorporate data set specific features (estimated from the training data set) via different methods and using them together would be redundant. One must also note that mapping results in modifying the video match score, while character-specific thresholding does not.

4.4.3.2 Understanding the improvement in the end-to-end character recognition accuracy due to thresholding

Key Observations

- *When all the characters were forwarded to the AVC, the improvement in the*

recognition accuracy for the subset of characters incorrectly recognized by the VTR is negated by a bigger degradation in the recognition accuracy for the subset that was correctly recognized by the VTR, resulting in a net degradation in the end-to-end character recognition accuracy.

- *In the setup used in this experiment, our system achieves an absolute improvement of 7.0% in the value of $\alpha(S)$ by correcting a number of ambiguous characters that were recognized incorrectly by the VTR and also suffers an absolute degradation of 1.6% in the value of $\alpha(S)$ for characters that were recognized correctly by the VTR but were designated as ambiguous by the thresholding component. The final $\alpha(S)$ value, with a net improvement of 5.4%, was 67.6%*

- *For the same setup, an ideal thresholding technique that is capable of accurately identifying the set of correctly and incorrectly recognized characters from the VTR output and designating them as non-ambiguous and ambiguous respectively, is shown to achieve a maximum $\alpha(S)$ value of 70.7% (compared to 67.6% for our sample thresholding technique).*

Explanation. This experiment was designed to study the reasons that contribute to the improvement or degradation in the recognition accuracy $\alpha(S)$ achieved by our audio-video based recognition system. We start by partitioning the test data set into four subsets TP, FN, TN and FP based on the absolute threshold ($AbsThr$=0.98), and the output of the VTR:

- *True Positives (TP)*: VTR output correct, designated non-ambiguous. Only VTR.
- *False Negatives (FN)*: VTR output correct, designated ambiguous. AVC.
- *True Negatives (TN)*: VTR output incorrect, designated ambiguous. AVC.
- *False Positives (FP)*: VTR output incorrect, designated non-ambiguous. Only VTR.

Table 6 shows the percentage of characters from the data set `DS-TEST-1` that fall into one of the four categories. While 46.1% characters were classified as TP, 16.1%

Table 6: Understanding thresholding using the True Positives, False Positives, True Negatives and False Negatives

System	True Positives $\frac{\lvert TP\rvert}{\lvert S\rvert} = 46.1\%$		False Negatives $\frac{\lvert FN\rvert}{\lvert S\rvert} = 16.1\%$		True Negatives $\frac{\lvert TN\rvert}{\lvert S\rvert} = 29.8\%$		False Positives $\frac{\lvert FP\rvert}{\lvert S\rvert} = 8.1\%$		Total Acc. $\alpha(S)$
	$\frac{\lvert TP_C\rvert}{\lvert TP\rvert}$ (%)	$\frac{\lvert TP_C\rvert}{\lvert S\rvert}$ (%)	$\frac{\lvert FN_C\rvert}{\lvert FN\rvert}$ (%)	$\frac{\lvert FN_C\rvert}{\lvert S\rvert}$ (%)	$\frac{\lvert TN_C\rvert}{\lvert TN\rvert}$ (%)	$\frac{\lvert TN_C\rvert}{\lvert S\rvert}$ (%)	$\frac{\lvert FP_C\rvert}{\lvert FP\rvert}$ (%)	$\frac{\lvert FP_C\rvert}{\lvert S\rvert}$ (%)	(%)
Baseline VTR[1]	100.0	46.1	100.0	16.1	0.0	0.0	0.0	0.0	62.2
All AVC[1,2]	83.4	38.4	89.7	14.5	23.6	7.0	17.9	1.4	61.4
Our System[1,2] (AbsThr=0.98)	100.0	46.1	89.7	14.5	23.6	7.0	0.0	0.0	67.6
Ideal System[1,2] (FN->VTR,FP->AVC)	100.0	46.1	100.0	16.1	23.6	7.0	17.9	1.4	70.7

Setup[1] - Mapping: Yes. **Setup**[2] - Option Selection: NumOpt=4, Synchronization: Rank Sum, Combination: Weighted (w_V=0.8,w_A=0.2).

are FN, 29.8% are TN and 8.1% are FP. For comparison, the character recognition accuracy for each of these four subsets was studied for different setups of the recognition system, which are described below:

Baseline VTR. The baseline VTR system is equivalent to a system whose thresholding component designates all characters in S as non-ambiguous. The end-to-end character recognition accuracy αS for such system was found to be 62.2%. Here, the subsets TP and FN are recognized with 100% accuracy and the subsets TN and FP are recognized with 0% accuracy.

All AVC. The all AVC system is equivalent to a system whose thresholding component designates all characters in S as ambiguous. This system is quite similar to the setup reported in row 4 of Table 4, except for the fact that our setup uses option selection with $NumOpt$=4. The degradation in the recognition accuracy of the full test

data set ($\alpha(S)$=61.4%) as compared to the baseline VTR serves as the motivation for thresholding. Notice that although we see an absolute improvement, as compared to baseline VTR, in the recognition accuracy for the subsets TN (7.0%) and FP (1.4%), this is offest by the larger degradation in the recognition accuracies for subsets TP and FN.

Our System Vs VTR. Our system has the same absolute accuracy values as the VTR for the TP (46.1%) and FP (0.0%) since these subset correspond to characters that have been designated as non-ambiguous and are therefore the output of the VTR only. The main advantage of using our system as opposed to the VTR stems from the fact that for the subset TN, which corresponds to the characters that were recognized incrorrectly by the VTR and were designated as ambiguous, our system was able to correctly recognize about 23.6% of them (7.0% absolute). At the same time, we observe a deterioration in the recognition accuracy for the subset FN, which corresponds to the characters that were correctly recognized by the VTR (100.0% accuracy) but were designated as ambiguous and forwarded to the AVC, which was able to correctly recognize only 89.7% (an absolute decrease of 1.6%, *i.e.* 16.1%-14.5%) of the character in FN. The final recogniztion accuracy $\alpha(S)$ for our system was found to be 67.6%.

Our System Vs All AVC. The main advantage of using our thresholding technique as opposed to a system where all the characters are forwarded to the AVC can been seen for the subset TP. Our system was able to recognize these characters (TP) with 100.0% accuracy since these only pass through the VTR as opposed to the the AVC system where the characters in TP are forwarded for audio-video based combination, which in this case, is only able to achieve an accuracy of 83.4% for the subset.

Our System Vs Ideal System. An ideal thresholding technique is the one that is capable of accurately identifying correct and incorrect output from the VTR. A recognition system with such a thresholding technique would be capable of accurately

identifying the characters in subsets TP and FN (*i.e.* the characters that were correctly identified by the VTR) and will designate them as non-ambiguous. Thus, such a system would achieve 100.0% accuracy for the subsets TP and FN. For the subsets TN and FP (*i.e.* the characters that were incorrectly recognized by the VTR), the ideal thresholding system will tag them as ambiguous and forward them to the AVC for audio-video based combination. As computed for the all AVC system, the recognition accuracy of the AVC for these two subsets is 23.6% and 17.9%. These low accuracy values may be caused by limitations of the mapping technique, the option selection technique, the synchronization, combination techniques and/or the audio text recognizer. A character may be incorrectly recognized if any one of these components does not pass the correct option to the next component or generates a low match score for the correct option. So given the same setup of the system (all other components other than the thresholding component) and an ideal thresholding technique, the maximum achievable character recognition accuracy $\alpha(S)$ was found to be 70.7%.

4.4.3.3 Effect of varying the thresholding parameters AbsThr, RelThr and $\alpha_F^{N'}$ on the end-to-end character recognition accuracy $\alpha(S)$

Key Observations

- *The values of the absolute threshold AbsThr and the relative threshold RelThr are varied and the effect on the end-to-end character recognition accuracy $\alpha(S)$ has been studied for a specific end-to-end setup of the system.*
- *Since the character specific thresholds depend on the parameter $\alpha_F^{N'}$, we vary the parameter $\alpha_F^{N'}$ and observe the effect on $\alpha(S)$.*
- *The maximum value of $\alpha(S)$ depends not only on the recognition accuracy of non-ambiguous characters $\alpha_V(S - S_D)$ and that of the ambiguous characters*

Table 7: Effect of varying the thresholding parameter *AbsThr* on the end-to-end character recognition accuracy $\alpha(S)$

Thresholding Parameter AbsThr	**Non. Amb. Chars.**		**Amb. Chars.**		**All Chars.**
	$\|S-S_D\|/\|S\|$	$\alpha_V(S-S_D)$ (%)	$\|S_D\|/\|S\|$	$\alpha_Z(S_D)$ (%)	$\alpha(S)$ (%)
`Baseline VTR`	1.00	62.2	0.00	0.0	62.2
`All AVC`	0.00	0.0	1.00	61.4	61.4
0.20	0.97	64.3	0.03	18.8	62.8
0.30	0.96	64.6	0.04	33.3	63.4
0.40	0.95	65.3	0.05	34.8	63.8
0.50	0.95	65.3	0.05	34.8	63.8
0.60	0.95	65.3	0.05	34.8	63.8
0.70	0.94	65.9	0.06	37.0	64.3
0.80	0.93	66.3	0.07	38.9	64.3
0.90	0.85	68.4	0.15	41.7	64.5
0.95	0.77	72.9	0.23	38.7	65.1
0.97	0.68	78.7	0.32	41.7	66.7
0.98	0.54	85.1	0.46	46.8	67.6
0.99	0.32	90.8	0.68	53.5	65.3

Setup - Mapping: `Yes`, Thresholding: `AbsThr=varying`, Option Selection: `NumOpt=4`, Synchronization: `Rank Sum`, Combination: `Weighted (`w_V`=0.8,`w_A`=0.2)`

$\alpha_Z(S_D)$, but also on the ratio of characters that are non-ambiguous $|S-S_D|/|S|$ and ambiguous $|S_D|/|S|$.

Explanation. We study how the end-to-end character recognition accuracy $\alpha(S)$ varies for each of the thresholding techniques namely absolute thresholding, relative thresholding and character-specific thresholding, by varying the values of their respective parameters *i.e.* *AbsThr*, *RelThr* and $\alpha_F^{N'}$.

Effect of varying AbsThr on $\alpha(S)$. Table 7 shows the effect of varying the absolute threshold *AbsThr* value on the ratio of non-ambiguous chars $|S-S_D|/|S|$, the VTR accuracy for the non-ambiguous characters $\alpha_V(S-S_D)$, the ratio of ambiguous characters $|S_D|/|S|$, the AVC accuracy for the ambiguous characters $\alpha_Z(S_D)$ and the end-to-end character recognition accuracy $\alpha(S)$. We notice that as the value of the

absolute threshold $AbsThr$ was increased, lesser number of characters were tagged as non-ambiguous and there was an increase in the accuracy value $\alpha_V(S-S_D)$ associated with such non-ambiguous characters. This can be attributed to the fact that for high values of the absolute threshold, the set of non-ambiguous characters contains only those characters that have a very high video match score for the first video option and therefore have a very high chance of already being correct (*i.e.* correctly recognized by the VTR). This results in a very high value of $\alpha_V(S-S_D)$ for such non-ambiguous characters (about 90.0% for $AbsThr$=0.99 in the reported experiment). We also notice that the accuracy for characters designated as ambiguous $\alpha_Z(S_D)$ also increased with an increase in $AbsThr$. Although the accuracy of the non-ambiguous characters $\alpha_V(S-S_D)$ and the accuracy of ambiguous characters $\alpha_Z(S_D)$ were found to be highest for $AbsThr$=0.99, the value of the end-to-end system accuracy $\alpha(S)$ was highest for $AbsThr$=0.98. This is due to the fact that the end-to-end character recognition accuracy $\alpha(S)$ is the weighted sum of the $\alpha_V(S-S_D)$ and $\alpha_Z(S_D)$ values, with weights of $|S-S_D|/|S|$ and $|S_D|/|S|$ respectively. For $AbsThr$=0.99, for instance, a much smaller fraction (0.32) of the characters were designated as non-ambiguous and were recognized with an accuracy of $\alpha_V(S-S_D)$=90.8% and a larger fraction (0.68) were recognized with an accuracy of $\alpha_Z(S_D)$=53.5%.

Effect of varying RelThr on $\alpha(S)$. Table 8 shows the effect of varying the relative threshold $RelThr$ on the same set of values as the previous experiment, *i.e.* $|S-S_D|/|S|$, $|S_D|/|S|$, $\alpha_V(S-S_D)$, $\alpha_Z(S_D)$ and $\alpha(S)$. The experiment, except for the use of relative thresholding technique, was configured in the same way as the absolute thresholding experiment reported above. Also, similar to the absolute thresholding experiment, there was a specific value of $RelThr$=0.97 that maximized the end-to-end character recognition accuracy $\alpha(S)$. The maximum $\alpha(S)$ value for the end-to-end character recognition accuracy was found to be 65.9% for this experiment. Notice that with relative thresholding, the number of ambiguous characters $|S_D|/|S|$

Table 8: Effect of varying the thresholding parameter $RelThr$ on the end-to-end character recognition accuracy $\alpha(S)$

Thresholding Parameter RelThr	**Non. Amb. Chars.**		**Amb. Chars.**		**All Chars.**
	$\|S - S_D\|/\|S\|$	$\alpha_V(S - S_D)$ (%)	$\|S_D\|/\|S\|$	$\alpha_Z(S_D)$ (%)	$\alpha(S)$ (%)
`Baseline VTR`	1.00	62.2	0.00	0.0	62.2
`All AVC`	0.00	0.0	1.00	64.0	64.0
0.80	0.32	74.4	0.68	59.5	64.3
0.85	0.40	75.5	0.60	56.6	64.3
0.90	0.55	73.5	0.45	51.9	63.8
0.95	0.65	70.9	0.35	51.2	64.0
0.97	0.85	67.8	0.15	54.9	65.9
0.99	0.95	65.3	0.05	56.5	64.9

Setup - Mapping: `Yes`, Thresholding: `RelThr=varying`, Option Selection: `OptRelThr=0.8`, Synchronization: `Rank Sum`, Combination: `Weighted (`w_V`=0.8,`w_A`=0.2)`

decreases with an increase in $RelThr$. This can be attributed to the condition for ambiguity detection using relative thresholds (based on the definition in Equation 16), which is based on the premise that correcty recognized charaters tend to have a lower value for the ratio between the video match scores of the second and first option.

Effect of varying $\alpha_F^{N'}$ on $\alpha(S)$. Since we cannot control the character-specific thresholds $T(c)$ (like we controlled the values of the absolute threshold $AbsThr$ and the relative threshold $RelThr$ above), we chose to vary the value of $\alpha_F^{N'}$ and study its effect on the end-to-end character recognition accuracy $\alpha(S)$. Note that $\alpha_F^{N'}$, here, represents an estimate of the character recognition accuracy of the audio-video based combination component in a scenario where all the input characters to the combination component have N video options out of which one is correct. When the value of the parameter $\alpha_F^{N'}$ is varied, it leads to different values for character-specific thresholds $T(c)$ that are estimated from the training set `DS-TRAIN-1` using the formulation derived in Section 4.2.2.2. Table 9 shows that the end-to-character

Table 9: Effect of varying the thresholding parameter $\alpha_F^{N'}$ on the end-to-end character recognition accuracy $\alpha(S)$

Thresholding Parameter $\alpha_F^{N'}$ (%)	Non. Amb. Chars.		Amb. Chars.		All Chars.
	$\|S - S_D\|/\|S\|$	$\alpha_V(S - S_D)$ (%)	$\|S_D\|/\|S\|$	$\alpha_Z(S_D)$ (%)	$\alpha(S)$ (%)
`Baseline VTR`	1.00	53.7	0.00	0.0	53.7
`All AVC`	0.00	0.0	1.00	51.0	51.0
30	0.81	67.8	0.19	32.3	61.0
40	0.78	69.1	0.22	36.2	62.0
50	0.76	71.0	0.24	34.8	62.2
60	0.77	70.0	0.23	34.2	61.8
80	0.76	70.5	0.24	33.0	61.6

Setup - Mapping: `No`, Thresholding: $\alpha_F^{N'}$=`varying`, Option Selection: `NumOpt=4`, Synchronization: `Rank Sum`, Combination: `Weighted (`w_V`=0.6,`w_A`=0.4)`

recognition accuracy $\alpha(S)$ was found to be the highest when $\alpha_F^{N'}$ values were between 40% to 50%. This can be attributed to the fact that the real value of $\alpha_F^{N'}$ for N=4 lies in the same range (this was verified experimentally for the test set `DS-TEST-1`) and therefore, there was a good match between the character-specific thresholds $T(c)$ generated using that value of $\alpha_F^{N'}$ and the test set scenario. It is difficult for us to estimate a single best value for $\alpha_F^{N'}$ for a given data set as it varies significantly for different characters and the performance of the synchronization and combination components also varies significantly for different characters. It may be interesting to investigate the use of character specific $\alpha_F^{N'}$ values as part of the future work. Unlike the absolute thresholding and relative thresholding cases, we cannot see the trend in the variation of the fractions of ambiguous and non-abiguous characters as well as their respective recognition accuracies as there is a different threshold for each character and the net effect of all these thresholds on the fraction of ambiguous characters may not be very clear. We have chosen to not use the setup of the recognition system with mapping because mapping and character-specific thresholding both perform a similar

operation. While mapping rescores the video match scores based on the training data, character-specific thresholding estimates suitable character-specific threshold values based on the training data. Interestingly, the maximum value achieved using character-specific thresholds for the experiment was 62.2% which happens to be the same value as the baseline VTR with mapping.

4.4.4 Option Selection Techniques

This section examines the option selection techniques proposed in this chapter. We start by providing the motivation for the option selection techniques by demonstrating the improvement in end-to-end character recognition accuracy that can be achieved by making use of our option selection techniques, compared to the baselines established earlier. We also conducted experiments to show the effect of different values of number of options and different absolute and relative option selection thresholds on the end-to-end character recognition accuracy.

4.4.4.1 Motivation for Option Selection

Key Observations

- *Even without thresholding, options selection techniques are effective at improving the end-to-end character recognition accuracy when compared to a setup where all the video options are forwarded for combination, both without and with mapping.*

Explanation. Table 10 reports experiments that were conducted to demonstrate the improvement in recognition accuracy $\alpha(S)$ that can be achieved by making use of the proposed option selection techniques, compared to the setup when all the segmented characters and all the video options are forwarded for combination. We conducted a set of experiments (reported in rows 2-8) to examine the effect of option selection on the accuracy $\alpha(S)$ using a setup where all the segmented characters are forwarded for

Table 10: Motivation for option selection

S.No.	Mapping	Thresholding	Option Selection	Accuracy (%) $\alpha(S)$
Effect of option selection on the accuracy $\alpha(S)$ without mapping[1]				
1.	No	No (all AVC)	No (all options)	50.0
2.	No	No (all AVC)	NumOpt=3	53.1
3.	No	No (all AVC)	OptAbsThr=0.80	55.0
4.	No	No (all AVC)	OptRelThr=0.90	56.0
Effect of option selection on the accuracy $\alpha(S)$ with mapping[2]				
5.	Yes	No (all AVC)	No (all options)	61.4
6.	Yes	No (all AVC)	NumOpt=3	61.6
7.	Yes	No (all AVC)	OptAbsThr=0.80	62.0
8.	Yes	No (all AVC)	OptRelThr=0.80	64.0

Setup[1] - Synchronization: `Rank Sum`, Combination: `Weighted (`w_V`=0.6,`w_A`=0.4)`
Setup[2] - Synchronization: `Rank Sum`, Combination: `Weighted (`w_V`=0.8,`w_A`=0.2)`

combination. In the experiments reported in the above mentioned rows, although, all the characters undergo recognition based on audio-video combination, option selection techniques limit the video options that are made available for combination with the output from audio text recognizer. Option selection, as expected, has the desirable effect of improving the recognition accuracy $\alpha(S)$ as compared to the scenarios (shown in rows 1 and 5) when all the segmented characters and all the video options are forwarded for audio-video based combination.

4.4.4.2 Effect of varying the option selection parameters NumOpt, OptAbsThr and OptRelThr on the end-to-end character recognition accuracy $\alpha(S)$

Key Observations

- *The effect of varying the option selection parameters NumOpt, OptAbsThr and OptRelThr values on end-to-end character recognition accuracy $\alpha(S)$ has been studied, with a suitable setup for each experiment.*
- *For a fixed thresholding technique and setup, varying the option selection parameters affects only the recognition accuracy of the ambiguous characters $\alpha_Z(S_D)$*

Table 11: Effect of varying the option selection parameter $NumOpt$ on the end-to-end character recognition accuracy $\alpha(S)$

Option Sel. Parameter NumOpt	**Non. Amb. Chars.**		**Amb. Chars.**		**All Chars.**
	$\|S-S_D\|/\|S\|$	$\alpha_V(S-S_D)$ (%)	$\|S_D\|/\|S\|$	$\alpha_Z(S_D)$ (%)	$\alpha(S)$ (%)
(VTR)1	0.66	71.3	0.34	19.5	53.7
2	"	"	"	34.8	58.9
3	"	"	"	36.6	59.5
4	"	"	"	37.2	59.7
5	"	"	"	34.8	58.9
6	"	"	"	30.5	57.4
7	"	"	"	26.8	56.2
8	"	"	"	23.8	55.2
10	"	"	"	23.2	55.0
20	"	"	"	23.2	55.0
(All Options)	"	"	"	23.2	55.0

Setup - Mapping: No, Thresholding: AbsThr=0.90, Option Selection: NumOpt=varying, Synchronization: Rank Sum, Combination: Weighted (w_V=0.6,w_A=0.4)

and does not affect the ratio of non-ambiguous characters $|S-S_D|/|S|$, the ratio of ambiguous characters $|S_D|/|S|$ and the character recognition accuracy of the non-ambiguous characters $\alpha_V(S-S_D)$.

Explanation. We conducted experiments to study the variation in the end-to-end character recognition accuracy $\alpha(S)$ for the variation in the value of the parameters associated with the option selection techniques proposed in this chapter. The experiments discussed below correspond to the proposed option selection techniques based on the number of options $NumOpt$, absolute threshold $OptAbsThr$ and relative threshold $OptRelThr$.

Effect of varying NumOpt on $\alpha(S)$. One option selection technique, as mentioned in Section 4.3, involves forwarding the top $NumOpt$ number of options to the A/V synchronization and combination components. Table 11 shows the effect of varying the number of options $NumOpt$ on the accuracy for the set of ambiguous

characters $\alpha_Z(S_D)$ and the end-to-end character recognition accuracy $\alpha(S)$. Since the same absolute thresholding technique, with the same value for $AbsThr$ was used for all the experiments, the value for the fraction of characters designated as non-ambiguous $|S - S_D|/|S|$=0.66, the fraction of ambiguous characters $|S_D|/|S|$=0.34 and the character recognition accuracy for the non-ambiguous characters $\alpha_V(S - S_D)$=71.3% remain constant for these experiments. The first row of the table, where $NumOpt$=1, corresponds to the setup that is equivalent to the VTR. In the experimental setup corresponding to row 1, although the ambiguous characters were forwarded for synchronization and combination, only the top video option (VTR output) was included with such ambiguous characters. As a result, the output with $NumOpt$=1 was the same as the output of the VTR. The last row of the table shows the setup where all the options were passed to the synchronization and the combination components and this is equivalent to a system without any option selection. These two extreme setups, when we forwarded 1 option and when we forwarded all the options, have an $\alpha(S)$ value of 53.7% and 55.0%, respectively. We also observed that the value of $\alpha(S)$ increases as the value of $NumOpt$ is increased from 1 to 4 and decreases for any further increase in the value of $NumOpt$. The maximum value for $\alpha(S)$=59.7% was achieved for $NumOpt$=4. The decrease in the value of $\alpha(S)$ for $NumOpt$>4 can be attributed to the fact that when too many video options are forwarded to the synchronization component, there is high likelihood that one of the incorrect video option may have an actual audio occurrence in the neighborhood, which makes this incorrect video option a good candidate for the recognized output. Note that the value of $\alpha(S)$ does not change for $NumOpt \geq 10$. This, in part, can be attributed to the specific weights (w_V=0.6,w_A=0.4) of the weighted combination technique and the low average video match scores associated with lower ranked video options (without mapping, the average video match score for the 10^{th} video option was found to be 0.66, as opposed to 0.93 for the top video option), which makes it hard for the 10^{th}

Table 12: Effect of varying the option selection parameter *OptAbsThr* on the end-to-end character recognition accuracy $\alpha(S)$

Option Sel. Parameter OptAbsThr	**Non. Amb. Chars.**		**Amb. Chars.**		**All Chars.**
	$\|S-S_D\|/\|S\|$	$\alpha_V(S-S_D)$ (%)	$\|S_D\|/\|S\|$	$\alpha_Z(S_D)$ (%)	$\alpha(S)$ (%)
`(All Options)` 0.00	0.66	71.3	0.34	23.2	55.0
0.20	”	”	”	23.2	55.0
0.40	”	”	”	23.2	55.0
0.60	”	”	”	27.4	56.4
0.80	”	”	”	36.0	59.3
0.85	”	”	”	37.2	59.7
0.86	”	”	”	34.1	58.7
0.87	”	”	”	24.4	55.4
`(VTR)` 1.00	”	”	”	19.5	53.7

Setup - Mapping: `No`, Thresholding: `AbsThr=0.90`, Option Selection: `OptAbsThr=varying`, Synchronization: `Rank Sum`, Combination: `Weighted (`w_V`=0.6,`w_A`=0.4)`

video option to become the top option after combination.

Effect of varying OptAbsThr on $\alpha(S)$. The absolute threshold based option selection technique, discussed in Section 4.3, forwards any video option that has a video match score greater than *OptAbsThr* to the A/V synchronization and combination components. Table 12 shows the effect of varying the absolute option selection threshold *OptAbsThr* on the accuracy $\alpha_Z(S_D)$ and the accuracy $\alpha(S)$. Again, since the same absolute thresholding technique, with same value for *AbsThr*=0.90 was used for all the experiments, the values for $|S-S_D|/|S|$=0.66, $|S_D|/|S|$=0.34 and $\alpha_V(S-S_D)$=71.3% remain constant for these experiments. Here, the setup of the recognition system shown corresponding to the first row of the table, with *OptAbsThr*=0.00, is equivalent to the setup where all the video options are forwarded to the synchronization and the combination components (except for the video options with video match score of 0.00). The setup shown in the last row of the table with *OptAbsThr*=1.00 produces the same output as the VTR, as the option selection component has been

Table 13: Effect of varying the option selection parameter *OptRelThr* on the end-to-end character recognition accuracy $\alpha(S)$

Option Sel. Parameter OptRelThr	**Non. Amb. Chars.**		**Amb. Chars.**		**All Chars.**
	$\|S-S_D\|/\|S\|$	$\alpha_V(S-S_D)$ (%)	$\|S_D\|/\|S\|$	$\alpha_Z(S_D)$ (%)	$\alpha(S)$ (%)
`(All Options)` 0.00	0.60	71.8	0.40	34.2	56.8
0.20	”	”	”	34.2	56.8
0.40	”	”	”	34.2	56.8
0.60	”	”	”	34.2	56.8
0.70	”	”	”	35.2	57.2
0.80	”	”	”	41.5	59.7
0.90	”	”	”	44.0	60.7
0.95	”	”	”	38.9	58.7
0.98	”	”	”	33.2	56.4
`(VTR)` 1.00	”	”	”	28.5	54.5

Setup - Mapping: `No`, Thresholding: `RelThr=0.95`, Option Selection: `OptRelThr=varying`, Synchronization: `Rank Sum`, Combination: `Weighted (`w_V`=0.6,`w_A`=0.4)`

implemented to forward, by default, the top video option to the subsequent components if no option is selected (note that there are no video options with video match score greater than 1.00) as a result of option selection. We observed that the both $\alpha_Z(S_D)$ and $\alpha(S)$ increase with increasing $OptAbsThr$, to reach a maximum value of $\alpha(S)$=59.7% for $OptAbsThr$=0.85 and then decrease for any further increase in the value of the threshold.

Effect of varying OptRelThr on $\alpha(S)$. The relative threshold based option selection technique, discussed in Section 4.3, forwards, to the synchronization and combination components, any video option $\mathbf{v}_i(s)$ that has $v_i^p(s)/v_1^p(s) > OptRelThr$. Similar to the previous experiments, Table 13 shows the effect of varying the relative option selection threshold $OptRelThr$ on the accuracy $\alpha_Z(S_D)$ and the accuracy $\alpha(S)$. Since the same relative thresholding technique, with same value for $RelThr$ was used for all the experiments, the values for $|S-S_D|/|S|$=0.60, $|S_D|/|S|$=0.40 and $\alpha_V(S-S_D)$=71.8% remain constant for these experiments. Here, the setup

of the recognition system shown corresponding to the first row of the table, with $OptRelThr$=0.00, is equivalent to the setup where all the video options are forwarded to the synchronization and the combination components (except for the video options with video match score of 0.00). The setup shown in the last row of the table with $OptRelThr$=1.00 produces the same output as the VTR, as the option selection component has been implemented to forward, by default, the top video option to the subsequent components if no option is selected as a result of option selection (note that for any character, the ratio of any video option's match score and the first video option's match score can never be greater than 1.00 since the first video option's match score is always greater than or equal to that of the other video options). We observed that both $\alpha_Z(S_D)$ and $\alpha(S)$ increase with increasing $OptRelThr$, to reach a maximum value of $\alpha(S)$=60.7% for $OptRelThr$=0.90 and then decrease for any further increase in the value of the threshold.

4.4.5 Consolidated Results

Table 14 shows a consolidated version of the baseline accuracies as well as the improvements in the end-to-end character recognition accuracy $\alpha(S)$ caused by mapping, thresholding and option selection. Note that the results here do not represent the maximum values obtained for each of the techniques and are not intended to demonstrate the superiority of any particular configuration of components over another. They are intended to motivate the need for mapping, thresholding and option selection and demonstrate that a suitable combination of all three can lead to a considerable improvement in the end-to-end character recognition accuracy $\alpha(S)$.

Baseline Accuracies and Mapping. The accuracy of the VTR without mapping, thresholding and option selection (in row 1a) serves as the primary baseline. The need for thresholding and option selection can be seen in row 1b where we see the degradation in $\alpha(S)$ when all the characters are passed to the AVC. Apart from

Table 14: Consolidated results showing the baselines and the improvement due to mapping, thresholding and option selection

S.No.	Mapping	Thresholding	Option Selection	Accuracy (%) $\alpha(S)$
Effect of thresholding and option selection on $\alpha(S)$ without mapping[1]				
1a.	No	No (only VTR)	No (n/a)	53.7
1b.	No	No (all AVC)	No (all options)	50.0
2a.	No	AbsThr=0.90	No (all options)	55.0
2b.	No	No (all AVC)	OptAbsThr=0.80	55.0
2c.	No	AbsThr=0.90	OptAbsThr=0.80	59.3
3a.	No	RelThr=0.95	No (all options)	56.8
3b.	No	No (all AVC)	OptRelThr=0.90	56.0
3c.	No	RelThr=0.95	OptRelThr=0.90	60.7
4a.	No	CharSpec ($\alpha_F^{N'}$=40)	No (all options)	61.8
4b.	No	No (all AVC)	NumOpt=4	51.0
4c.	No	CharSpec ($\alpha_F^{N'}$=40)	NumOpt=4	62.0
Effect of thresholding and option selection on $\alpha(S)$ with mapping[2]				
5a.	Yes	No (only VTR)	No (n/a)	62.2
5b.	Yes	No (all AVC)	No (all options)	61.4
6a.	Yes	AbsThr=0.95	No (all options)	63.8
6b.	Yes	No (all AVC)	OptAbsThr=0.80	62.0
6c.	Yes	AbsThr=0.95	OptAbsThr=0.80	64.7
7a.	Yes	RelThr=0.98	No (all options)	64.5
7b.	Yes	No (all AVC)	OptRelThr=0.80	64.0
7c.	Yes	RelThr=0.98	OptRelThr=0.80	65.9

Setup[1] - Synchronization: Rank Sum, Combination: Weighted (w_V=0.6,w_A=0.4)
Setup[2] - Synchronization: Rank Sum, Combination: Weighted (w_V=0.8,w_A=0.2)

the baseline VTR (row 1a), this serves as a useful reference for all the thresholding techniques. The improvement that mapping brings to the VTR and also the AVC can be seen in rows 5a-b with reference to 1a-b. Row 5a serves as a baseline for all the setups with mapping and row 5b serves as a another reference for the thresholding techniques (with mapping).

Thresholding. For the system without mapping, the improvement caused by using each of the thresholding techniques as opposed to passing all the characters through the AVC can be seen by comparing rows 2a, 3a and 4a with the same setup without

thresholding shown in row 1b. The same phenomenon can be seen for the system with mapping by comparing rows 6a and 7a with row 5b. Character-specific thresholding and mapping have not been used together since they are based on very similar measures that are computed for each character based on the training set although the mapping stage modifies the video match scores to generate a better input for the thresholding and option selection techniques.

Option Selection. By comparing rows 2b, 3b and 4b with row 1b, we can see the improvement caused by using each of the option selection techniques (in a system without mapping) instead of passing all the options to the AVC. When mapping is used, a similar trend is observed in rows 6b and 7b when compared to row 5b.

Using mapping, thresholding and option selection together. We can see from each of the sets of experiments 2a-c, 3a-c, 4a-c, 6a-c and 7a-c that using a suitable combination of thresholding and option selection techniques proves to be much better than using them separately. We also see the best end-to-end character recognition accuracy $\alpha(S)$ values for setups that make use of a combination of suitable mapping, thresholding and option selection techniques (such as those in rows 6c and 7c).

4.5 Summary

In this chapter we presented a set of techniques that could be used in preparation for the combination stage to reduce the errors introduced during combination and to improve the chances of correcting the errors in the video text recognizer output. Towards this end, we proposed a three step approach, which included mapping, thresholding and option selection. The mapping step, if enabled, rescores and reorders the video options generated by the video text recognizer based on the training data set and is therefore able to incorporate nuances specific to the data set. The thresholding step applies a common absolute or relative threshold for all the characters or it

makes use of character-specific thresholds, to designate the characters as ambiguous or non-ambiguous. For each segmented character that has been designated as being ambiguous, the option selection step determines the video options that should be forwarded for audio-video based synchronization and combination. We have proposed some option selection techniques that are based on a fixed number of options or the video match scores of the options. Experiments conducted over the dataset `DS-TEST-1` show that the proposed techniques, when utilized appropriately, can help in significantly improving the end-to-end character recognition accuracy.

CHAPTER V

AUDIO-VIDEO SYNCHRONIZATION

5.1 Introduction

Recognition systems that combine the output of multiple recognizers are often designed to identify the same underlying entity. For instance, multiple recognizers could be working together to recognize the same image of a human face, the same image of a handwritten character or to identify the same gesture in a video segment. In such scenarios, the common underlying entity establishes the correspondence between the output of the multiple recognizers, which is then used to combine the output from the recognizers. However, in scenarios where one is attempting to combine the output from multiple recognizers that do not operate on the same underlying entity (*e.g.* a handwriting recognizer operating on characters segmented from a classroom video and a speech recognizer operating on the spoken content from the same video), it must be first established that a specific output from each of the recognizers refers to the same entity (*e.g.* same character being written and spoken) before the output can be combined. Audio-video based recognition, as is evident, falls in the category where the output of the recognizers must be explicitly synchronized before their outputs can be combined. In this chapter, we discuss the challenges associated with audio-video synchronization in classroom videos and describe techniques that can be used to achieve the same.

The ability to correctly synchronize the output from the video and the audio text recognizers is critical to the accuracy of the end-to-end audio-video based recognition system [112]. However, due to issues like the shadows that affect the video timestamping accuracy, the occlusions caused by the instructor, the skew between the

writing and the utterance of a character and the errors in the output of the video and the audio text recognizers, establishing correspondence between the handwritten content and the spoken content can be hard. Experiments, reported later in this chapter, reveal that simple synchronization techniques that work well for a specific scenario (*e.g.* videos without shadows and occlusions) often fail for another scenario (*e.g.* videos with shadows and occlusions). We have developed a suite of audio-video synchronization techniques, consisting of both, simple techniques that are geared for a specific scenario, to more general techniques that perform well under a range of scenarios.

The remainder of this chapter is organized as follows. Section 5.2 revisits and extends the mathematical model for audio-video synchronization described earlier. Section 5.3 describes the factors that affect audio-video synchronization. The techniques for audio-video synchronization are described in Section 5.4. Section 5.5 presents a detailed evaluation of the techniques presented in this chapter. Finally, a summary of the findings is presented in Section 5.6.

5.2 Audio-Video Synchronization - Mathematical Model

In this section, we present the mathematical model associated with the audio-video (A/V) synchronization component of our end-to-end recognition system. The synchronization component receives its input from the ambiguity detection and the option selection component. The input consists of the set of ambiguous segmented characters S_D, each with a set of video options $O(s)$ that were selected by the option selection component for subsequent combination with the output of the audio text recognizer. For each video option $\mathbf{v}_j(s) \in O(s)$ of a given ambiguous segmented character $s \in S_D$, the A/V synchronization component, first, uses the audio text recognizer to determine a set of possible audio options $A_j(s)$. Next, from the set of

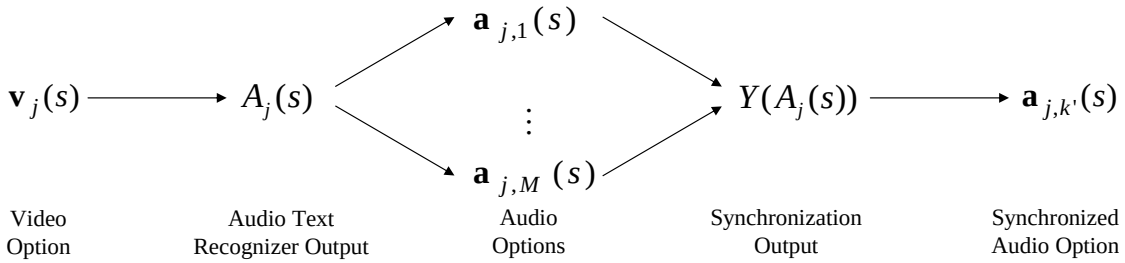


Figure 20: Showing a single video option, audio text recognizer output, audio options, synchronization output and the synchronized audio option

possible audio options $A_j(s)$, the A/V synchronization component determines a *suitable* audio option $\mathbf{a}_{j,k'}(s)$, which it considers to be best synchronized with the video option $\mathbf{v}_j(s) \in O(s)$. This is shown in Figure 20, and is represented as

$$Y(A_j(s)) = \mathbf{a}_{j,k'}(s) \tag{32}$$

where $Y(A_j(s))$ is the output of the synchronization component. Note that in some cases, where there is no suitable audio option for the supplied video option, k' may not exist.

As a result of A/V synchronization, each video option $\mathbf{v}_j(s) \in O(s)$ of a given segmented character $s \in S_D$ has at most one synchronized audio option. The A/V combination component, described in the following chapter, makes use of the set of synchronized video and audio options corresponding to each ambiguous character to arrive at the final character recognition output.

A/V Synchronization Accuracy. We make use of A/V synchronization accuracy, represented as $\alpha_Y(S_D)$, to evaluate the performance of the synchronization techniques proposed in this chapter. The synchronization accuracy is a measure of the ability of the proposed techniques to correctly locate, for a given segmented character, the audio option corresponding to the correct video option. Let $\mathbf{v}_g(s)$ represent the correct video option, *i.e.* $v_g^c(s) = G(s)$, from the set of video options $V(s)$ for a segmented character $s \in S_D$ and $A_g(s)$ be the set of audio options corresponding to

the video option $\mathbf{v}_g(s)$. A segmented character $s \in S_D$ is considered to be correctly synchronized if the absolute difference between the timestamp $a^t_{g,k'}(s)$ corresponding to the synchronized audio option $\mathbf{a}_{g,k'}(s)$ and the manual audio timestamp $TS^m_A(s)$ for the character s is less than or equal a given threshold τ_{sync}. The function $E_Y(s)$, defined below, produces 1 if the synchronization output is correct and produces 0 otherwise.

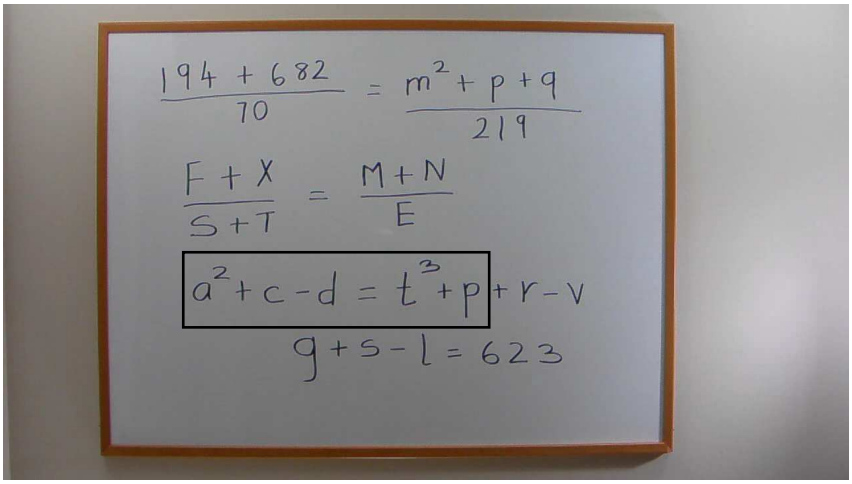
$$E_Y(s) = \begin{cases} 1, & \text{if } |a^t_{g,k'}(s) - TS^m_A(s)| \leq \tau_{sync} \\ 0, & \text{otherwise} \end{cases} \tag{33}$$

The synchronization accuracy can therefore be calculated as follows

$$\alpha_Y(S_D) = \frac{\sum\limits_{\forall s \in S_D} E_Y(s)}{|S_D|} \tag{34}$$

5.3 Factors Affecting A/V Synchronization

In the context of the audio-video based mathematical content recognition system proposed in this dissertation, synchronizing the video and the audio content is an important problem. The accuracy of synchronization is affected by several factors and understanding these factors helps in a better design of the synchronization techniques that take into account the nature of the input classroom video, the quality of the preprocessing and the accuracy of the recognizers that constitute the end-to-end recognition system. The remainder of this section, with the help of the example shown in Figure 21, describes the factors that affect synchronization. The impact of these factors has also been studied experimentally and the results are reported in Section 5.5.



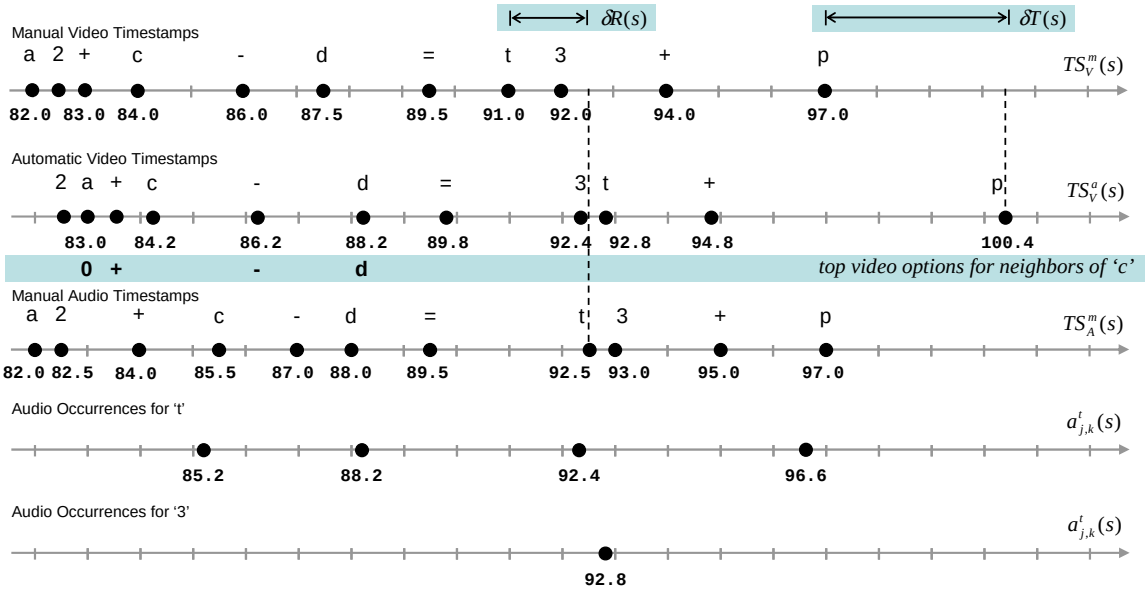


Figure 21: Factors affecting A/V synchronization

5.3.1 Video Timestamping

One would expect that for a video that is recorded such that the writing and the utterance of each of the characters is well aligned (*i.e.* the character is written and spoken at the same time), the audio-video synchronization should be fairly straightforward. However, during the automatic video timestamping, if the handwritten character is detected much after it is written, then the corresponding audio utterance is no longer aligned for that character, making the synchronization problem more

challenging. Our synchronization system relies heavily on the automatically determined (by the preprocessing stage) video timestamp, $TS_V^a(s)$ associated with each handwritten character s. The video timestamping is considered to be good if the timestamp $TS_V^a(s)$ is in close proximity of the manually determined video timestamp $TS_V^m(s)$. As mentioned earlier, the manual video timestamp roughly corresponds to the time instant when the character becomes fully visible in the classroom video. For each character, the timestamping difference $\delta T(s) = \mid TS_V^a(s) - TS_V^m(s) \mid$ is computed and this determines whether the character is considered to have a good timestamping or a poor timestamping. We have observed that the simple video timestamping technique that uses a single binarization threshold (described in Section 3.5.1) tends to have a much higher value for $\delta T(s)$ in regions that may have been impacted by the instructor's shadow. In some scenarios, the difference $\delta T(s)$ can be as large as 8-10 seconds, which can result in an incorrect synchronization even when the character may have been handwritten (and visible in the video) and spoken at the same time. The adaptive video timestamping (described in Section 3.5.4) makes use of two binarization thresholds for a given frame - one for the region impacted by the instructor's shadow and another one for the region without any shadow. The technique is able to significantly reduce the number of segmented characters that have $\delta T(s) > 2$ seconds. While the adaptive timestamping technique is able to achieve good timestamping accuracy for our data set, the same may not be true for videos in general and therefore, it is important to study the impact of the video timestamping quality on the synchronization accuracy. Experimental results, presented later, demonstrate that, in choosing the synchronization technique, one must take into account the accuracy of the video timestamping.

Figure 21 shows the various timestamps associated with a selected set of segmented characters. The selected set of segmented characters is shown enclosed in a black rectangle drawn on the frame of interest from the corresponding video. On each

of the axes (shown as horizontal lines) below the frame of interest, the black dot indicates the timestamp for a given character and the corresponding character name appears as a label above the dot. For instance, on the axis labeled $TS_V^m(s)$ the black dot at time t =97.0 seconds corresponds to the character `'p'`. The figure shows the timestamping difference $\delta T(s)$ for the character `'p'`, *i.e.* the time difference between the manual video timestamps $TS_V^m(s)$ and the automatic video timestamps $TS_V^a(s)$. The figure also shows that although the timestamping is reasonably good (with low $\delta T(s)$ values) for the majority of the characters, we do observe that in some cases it is higher, such as for the character `'p'`. Also notice that in some cases the order of the automatic timestamps changes compared to the manual ones, such as for characters `'t'` and `'3'`. The high $\delta T(s)$ value for the character `'p'` and the reordering, in this case, were caused by the instructor's shadow.

5.3.2 A/V Recording Alignment

In the discussion above, we showed that synchronization, in a given video, is not straightforward even when the writing and the utterance of each character from the video is well aligned. Classroom videos that do not have the writing and the utterance of the characters well aligned make the synchronization problem even harder. We consider the A/V recording alignment to be high when the average value of the recording time difference, $\delta R(s) =\mid TS_V^m(s) - TS_A^m(s) \mid$ is low. Here, $TS_A^m(s)$ is the manually determined audio timestamp and it corresponds to the time instant when the character s is spoken by the instructor. Occlusion is one factor that contributes to poor A/V recording alignment. To quantify the duration of occlusion, we define $TS_V^w(s)$ to represent the actual time instant when the character s is actually written (may not be visible in the video) on the whiteboard. Note that $TS_V^w(s)$ has been defined for purely theoretical understanding and due to the presence of occlusions, it cannot be determined by observing the video. The $\delta R(s)$ value depends on the skew

between the actual writing and the utterance ($\mid TS_V^w(s) - TS_A^m(s) \mid$) of the character s and is also dependent on the amount of occlusion, which introduces an additional delay of $\mid TS_V^w(s) - TS_V^m(s) \mid$, and can be mathematically represented as follows.

$$\mid TS_V^m(s) - TS_A^m(s) \mid = \mid\mid TS_V^w(s) - TS_A^m(s) \mid \pm \mid TS_V^w(s) - TS_V^m(s) \mid\mid \tag{35}$$

Unlike the timestamping difference $\delta T(s)$ which can be entirely attributed to the shadows caused by instructor and can therefore be eliminated by a better timestamping technique, the recording time difference $\delta R(s)$ is intrinsic to the way the video was recorded and cannot be eliminated by use of advanced preprocessing techniques. However, if the video was recorded without any occlusions and without any skew between the writing of the characters and their utterances, then the $\delta R(s)$ for all the characters will be negligible.

In Figure 21, the manual video timestamps $TS_V^m(s)$ and the manual audio timestamps $TS_A^m(s)$ are shown and the absolute difference between them *i.e.* the recording time difference $\delta R(s)$, is shown for character `'t'`. We observe that the $\delta R(s)$ values are very small for most of the characters and therefore the audio and video have good recording alignment. This video was recorded with minimal occlusions and without a significant skew between the writing of the characters and their utterance.

5.3.3 VTR Accuracy

The characters in the video that are in the temporal vicinity (as determined by the automatic video timestamps $TS_V^a(s)$) of the character under investigation are referred to as its *neighbors*. The audio options corresponding to such neighbors are assumed to be located in close proximity of the audio option of the character under investigation, and some of the proposed synchronization techniques use this as a feature to implement A/V synchronization. VTR accuracy plays an important role here, as the audio search term that is used to lookup the neighbors for the purpose

of synchronization is based on the top video option determined by the video text recognizer. If the accuracy of the video text recognizer is low, then the neighbors have a higher chance of being incorrectly recognized and therefore there is a higher chance that an incorrect audio search term is used for lookup. This impacts the correctness of the neighbor based feature used by the synchronization techniques, resulting in an incorrect A/V synchronization.

In the example shown in Figure 21, assume that we are synchronizing the character `'c'` by using the neighbors. If 2 video neighbors before (based on their $TS_V^a(s)$ values) and 2 after the character `'c'` are considered for the purpose of synchronization, then the characters `'a'`, `'+'`, `'-'` and `'d'` form the neighbors. However, we must note that since the VTR accuracy is not 100%, the neighbors may have been incorrectly recognized by the video text recognizer. The top video option for each of these neighbors is shown in the figure. Here, the character `'a'` has an incorrect top option, which was recognized as `'0'`. To compute the neighbor-related feature, the synchronization technique therefore searches for the utterance of `'0'` in the audio instead of the utterance of `'a'`. Such video text recognizer errors in the neighbors may lead to errors in synchronization especially when there is a higher ratio of incorrectly recognized neighbors.

5.3.4 ATR Accuracy

The audio text recognizer may cause false positives or false negatives, which can lead to incorrect A/V synchronization. Here, false positives refer to the hits generated by the audio text recognizer when the character has not been spoken. In general, such false positives are higher for characters with a smaller number of phonemes in the corresponding audio search term. False negatives refer to the case when the audio text recognizer is unable to find the audio occurrence of the character when the character has been spoken. The accuracy of the output of the audio text recognizer impacts

A/V synchronization in two ways, (1) for the character being synchronized any false positive or false negative generated by the audio text recognizer can lead to incorrect synchronization, and (2) when using neighbors for A/V synchronization, an incorrect output by the audio text recognizer can lead to an incorrect count for the number of audio occurrences of the neighbors in the vicinity of a candidate audio option.

We can see in Figure 21 that a search for the character `'t'` in the audio using the audio text recognizer, apart from generating the correct audio occurrence at 92.4 seconds, generates several false positives (at 85.2 seconds, 88.2 seconds and 96.6 seconds) as well. Such false positives may in turn lead to synchronization errors. These false positives can be attributed to the fact that the search term `'_t_iy'` corresponding to character `'t'` has a very small number of phonemes (2 phonemes) and there were utterances of other characters such as `'c'`, `'d'` and `'p'` which have a common phoneme of `'_iy'`. When we searched for the character `'3'` (audio search term: `'three'` and `'cube'`) in the audio, we got only 1 audio occurrence corresponding to the audio search term `'cube'` at 92.8 seconds (there were a few other occurrences with extremely low audio match scores that occur due to common phonemes between the audio search term `'three'` and the search term for characters `'c'`, `'d'`, `'t'` and `'p'`, but as we will see later that such audio options are pruned away by our synchronization techniques). The audio occurrence at 92.8 seconds corresponds to the real utterance of the character `'3'` and there are no false positives, making the synchronization task less challenging. This is due to the fact that the audio search term `'cube'` (phonetic equivalent: `'_k_y_uw_b'`) has 4 phonemes, which is not too low.

5.4 A/V Synchronization Techniques

In this section we start by describing a set of features that are calculated for each audio option. These audio features are used by the A/V synchronization techniques

proposed later in this section. As mentioned earlier, we assume that everything that is handwritten is spoken and we focus on handling synchronization issues that arise due to occlusions, shadows, the skew between the writing and the utterance of a character and also errors in the video and audio recognizer output.

5.4.1 Audio Features

For each audio option, $\mathbf{a}_{j,k}(s)$, four audio features are computed, $F_i(\mathbf{a}_{j,k}(s))$ for $i = 1, 2, 3, 4$. The first feature is the audio match score that is generated by the audio text recognizer for the audio option $\mathbf{a}_{j,k}(s)$,

$$F_1(\mathbf{a}_{j,k}(s)) = a^p_{j,k}(s) \tag{36}$$

The second feature is the difference between the automatically generated video timestamp and the automatically generated audio timestamp,

$$F_2(\mathbf{a}_{j,k}(s)) = |TS^a_V(s) - a^t_{j,k}(s)| \tag{37}$$

The third feature is defined in terms of the number of video options before and after $\mathbf{v}_j(s)$ that are found within a window of time around a given audio option $\mathbf{a}_{j,k}(s)$. More specifically, let $[W_B, W_A]$ be a time window around audio option $\mathbf{a}_{j,k}(s)$, and let N_B and N_A be integers. Feature $F_3(\mathbf{a}_{j,k}(s))$ is then defined as the number of audio options that are found within the time window $[W_B, W_A]$ that correspond to N_B video options *before* and N_A video options *after* video option $\mathbf{v}_j(s)$. As a specific example, suppose that written on the whiteboard is the equation

$$a^2 + b^2 = c^2$$

and let $\mathbf{a}_{j,k}(s)$ be an audio option for the segmented character `'b'`. If

$$[W_B, W_A] = [2, 2] \tag{38}$$

and $N_B = N_A = 2$, then F_3 will be the number of audio options that are found within a window two seconds before and two seconds after $\mathbf{a}_{j,k}(s)$ that correspond to the two characters before and the two characters after `'b'` on the whiteboard. In other words, out of the four "neighbors" of the character `'b'`, F_3 will be equal to the number of neighbors that are found in the audio stream within the given time window. Thus, in this case, F_3 may have any value between zero and four.

The fourth feature, F_4, differs from F_3 in that instead of counting how many of the N_B video options before $\mathbf{v}_j(s)$ and how many of the N_A video options after $\mathbf{v}_j(s)$ are found within a time window $[W_B, W_A]$ around $\mathbf{a}_{j,k}(s)$, here the window is expressed as $[n_B/t_B, n_A/t_A]$ and involves the selection of n_B out of t_B neighbors before the given video option $\mathbf{v}_j(s)$, and n_A out of t_A neighbors after $\mathbf{v}_j(s)$. For example, if

$$[n_B/t_B, n_A/t_A] = [2/3, 2/3] \tag{39}$$

then two out of three neighbors are selected before the video option and two out of three after it. The value of the feature F_4 depends on these selected neighbors. Techniques for selection of neighbors are described in Section 5.4.5.

5.4.2 Initial Pruning

The first step in each of the proposed audio-video synchronization techniques is to prune the audio options that are highly unlikely to be the final synchronized audio option. There are two pruning conditions that are based on F_1 and F_2. The first is to remove any audio option for which F_1 is less than some threshold, P_P. Thus, any audio option that has an audio match score that is less than P_P will be pruned. The second is to remove any audio option that has a value for F_2 that falls outside a given window, $[P_B, P_A]$. Since F_2 is the difference between the audio and video timestamps, then an audio option will be pruned if either of the following two conditions are met,

$$TS_A^a(\mathbf{a}_{j,k}(s)) > TS_V^a(s) + P_A \quad (40)$$

$$TS_A^a(\mathbf{a}_{j,k}(s)) < TS_V^a(s) - P_B \quad (41)$$

For example, a pruning window $[P_B, P_A] = [-8, 6]$ will remove those audio options that are found more than 8 seconds after or more than 6 seconds before the video option.

The pruning operation is represented as a function Q that operates on the set of audio options $A_j(s)$ that correspond to a video option $\mathbf{v}_j(s)$. The output of the pruning operation, $Q(A_j(s))$, is the set of audio options that remain after pruning.

5.4.3 Time Difference Based Technique

For every video option $\mathbf{v}_j(s)$, the time difference based technique is designed to take the set of pruned audio options $Q(A_j(s))$ as input and determine the synchronized audio option $\mathbf{a}_{j,k'}(s)$, where k' is the index of the audio option that is closest in time to the video option. Towards this end, the technique selects the audio option with the lowest value of the feature F_2. The index k' of the selected audio option can be determined as follows:

$$k' = \arg\min_k(F_2(\mathbf{a}_{j,k}(s))) \quad (42)$$

where $\mathbf{a}_{j,k}(s) \in Q(A_j(s))$. The audio option $\mathbf{a}_{j,k'}(s)$ is returned as the final synchronization output $Y(A_j(s))$.

Example. To better explain the time difference based synchronization technique and to understand the synchronization related issues we make use of an example. Figure 22 shows a sample video frame and the characters used in this example are those in the equation

$$a^2 + c - d = t^3 + p + r - v$$

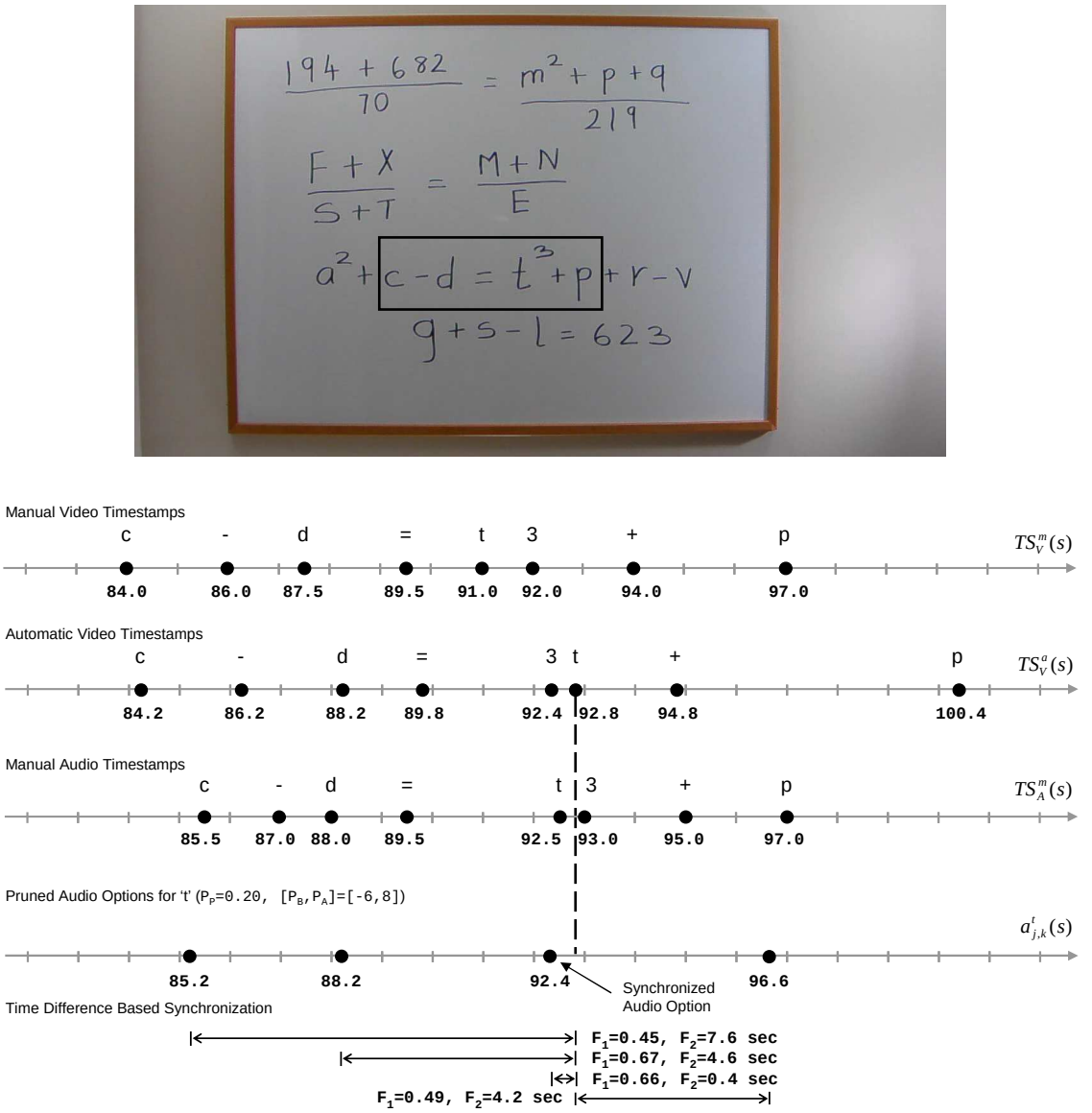


Figure 22: Example showing the working of the time difference based synchronization technique

The figure also shows the manual video timestamps $TS_V^m(s)$ and the manual audio timestamps $TS_A^m(s)$ of the characters to provide more information about the nature of the recorded video. As can be observed, the time difference between $TS_V^m(s)$ and $TS_A^m(s)$ for most characters is quite small (less than 2 seconds) and therefore the video can be considered to have a good recording alignment. Upon comparing the automatic video timestamps $TS_V^a(s)$ with the manual equivalents $TS_V^m(s)$, we

observe that in most cases (except for the character 'p') the video timestamping is fairly accurate. The character that we are considering here is the character 't' and we demonstrate the synchronization process for the correct video option $\mathbf{v}_g(s)$ (*i.e.* the video that corresponds to the correct output 't') as $\mathbf{v}_g(s)$ is the video option that contributes to the synchronization accuracy defined in Equation 34. We search for audio occurrences of the phonetic equivalent of 't' by using the audio text recognizer and then prune the set of audio options $A_g(s)$ using the parameter P_P=0.20 which operates on feature F_1 and the parameter $[P_B, P_A]$=[-6,8] which operates on feature F_2 to return a pruned set of audio options $Q(A_g(s))$. The audio options in the set $Q(A_g(s))$ are shown plotted on the time axis labeled $a^t_{g,k}(s)$. The task here is to select from the audio options in $Q(A_g(s))$, the audio option which corresponds to the video option for the character 't'. Time difference based synchronization selects the audio option from $Q(A_g(s))$ which has the lowest value for the feature F_2. In this case, the audio option at 92.4 seconds has the lowest $F2$ value of 0.4 seconds and has therefore been selected as the synchronization output, which is in fact the correct output. Here, the time difference based synchronization technique was able to perform correct synchronization since both the video timestamping and recording alignment are good for the video used in this example. By inspecting the manual audio timestamps $TS^m_A(s)$, we realize that the other audio options in $Q(A_g(s))$ are caused due to the presence of similar sounding utterances at that time, such as the characters 'c', 'd' and 'p' all of which have just 2 phonemes and one of them common with the character we have searched for *i.e.* 't'. This example also demonstrates how an audio search term with a smaller number of phonemes often results in an increased number of false positives making the synchronization task more challenging.

5.4.4 Neighbor Based Technique

With the neighbor-based synchronization technique, the audio option that is selected, $\mathbf{a}_{j,k'}(s)$, is the one that has the highest number of video neighbors of $\mathbf{v}_j(s)$ that are spoken within a time window centered around $a^t_{j,k'}(s)$. Thus, from the set of pruned options, the neighbor-based technique selects the audio option that has the largest value of F_3,

$$k' = \arg\max_k(F_3(a_{j,k}(s))) \tag{43}$$

Since the neighbor-based technique may result in a tie between two or more audio options, in such cases the tie is resolved by selecting the audio option that has the smallest value of the audio feature F_2.

Example. An example illustrating the neighbor-based synchronization technique is shown in Figure 23. The characters of interest are those in the equation

$$\frac{r+s}{319} = 70 + 469$$

The timelines below the whiteboard shows the various timestamps associated with the characters in this equation. Comparing the manually generated timestamps for the audio and video it is clear that the time at which the characters are spoken is close to the time at which they are written. However, comparing the manually generated video timestamps with those that are generated automatically, we see that for this video segment the automatic video timestamping performs poorly. Consider, for example, the character `'9'` that is written at time 62.0 seconds. With audio pruning parameters of $P_P = 0.20$, which only allows audio options with an audio score of 0.20 or higher, and $[P_B, P_A] = [-6, 16]$, which sets the only allowable audio options to those that are 6 seconds before and 16 seconds after the video option with $v^c_j(s) = 9$, we see that there are only two audio options in the pruned set, one at 62.2 seconds and one at

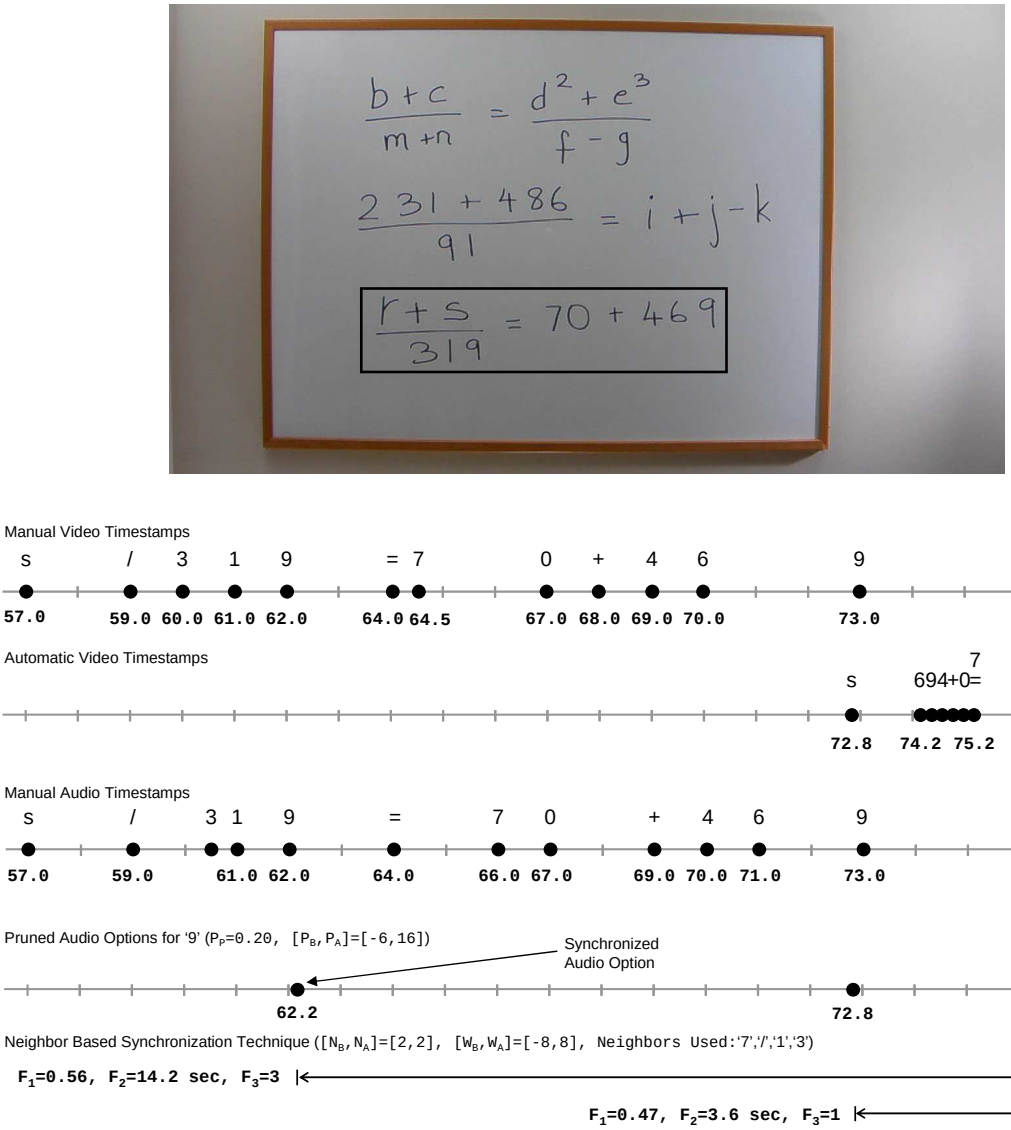


Figure 23: Example showing the working of the neighbor based synchronization technique

72.8 seconds. The values of the audio features F_1, F_2, and F_3 for the two pruned audio options are shown at the bottom of the figure. Note that value of F_2, the difference between the automatic audio and video timestamps, is smaller for the incorrect audio option making time-difference based synchronization unsuitable for this case. With $[N_B, N_A] = [2, 2]$, the video neighbors that are considered for computing F_3 are the characters `'7'`, `'/'`, `'1'` and `'3'`, which are the two characters before the `'9'` and the two characters after it, based on the automatic video timestamps, $TS_V^a(s)$, which

are shown in the second timeline in the figure. For each audio option, the character names for the top video option of each of these neighbors are searched for in the audio within a window $[W_B, W_A] = [-8, 8]$ that is centered on the times of the audio options, *i.e.*, at 62.2 seconds and 72.8 seconds. The value of feature F_3 for the audio option at time 62.2 seconds is 3 and it is equal to 1 for the audio option at time 72.8 seconds. Since F_3 is larger for the audio option at 62.2 seconds, this option selected by the neighbor-based synchronization technique. Note that even though all four neighbors are spoken within the window $[W_B, W_A]$ around the audio option at 62.2 seconds, the value of F_3 is equal to three instead of four. This is because the top video option of the neighbors is not always correct and for incorrectly recognized neighbors our technique will look for audio terms that do not exist in the window. Another reason why this may happen is that there may be false negatives *i.e.* the character, although it is spoken, is not detected by the audio text recognizer. This serves as the motivation for the selective-neighbor based technique, which is described in the following section.

5.4.5 Selective Neighbor Based Technique

In the neighbor based synchronization technique, the top video option of the neighbors is assumed to be the best guess of the final character recognition output and therefore, the audio search term corresponding to the top option's character name is used to locate neighbors in the audio component and determine the value of feature F_3. However, given the fact that in our set up the character recognition accuracy of the video text recognizer is about 50%, we can expect about half of these neighbors to be unreliable on account of being incorrectly recognized by the video text recognizer. Experimental results, presented later in this chapter, show that having a higher percentage of correctly recognized neighbors helps in synchronization. This motivates the need for selecting neighbors that have a higher probability of being correct for

the purpose of synchronization. Similar to the neighbor based technique the selective neighbor based technique is designed to return, for every video option $\mathbf{v}_j(s)$, an audio option $\mathbf{a}_{j,k'}(s) \in A_j(s)$ that has the highest number of 'selected' video neighbors of $\mathbf{v}_j(s)$ being uttered in its vicinity in the audio component. From the set of pruned options $Q(A_j(s))$, the neighbor based technique selects the audio option that has the highest value of F_4, which is considered to be the synchronization output $Y(A_j(s))$. The index k' for the selected audio option can be determined as follows:

$$k' = \arg\max_{k}(F_4(\mathbf{a}_{j,k}(s))) \tag{44}$$

where $\mathbf{a}_{j,k}(s) \in Q(A_j(s))$. The neighbor selection could also be driven by the accuracy of the audio text recognizer for various characters and one could select characters that are better recognized by the audio text recognizer. Therefore, the selection of suitable neighbors may be based on video features related to the video text recognizer accuracy or audio features related to the audio text recognizer accuracy, as explained below.

5.4.5.1 Video Based Neighbor Selection

Video based neighbor selection focuses on selecting the neighbors that have a higher chance of being correctly recognized (*i.e.* the top video option is correct) by the video text recognizer. Since ambiguity detection techniques, discussed in Chapter 4, were designed to detect those characters that have a higher chance of being incorrectly recognized (*i.e.* the set of ambiguous characters S_D) by the video text recognizer, we make use of some of the same ideas here to detect better neighbors.

Mapping. As shown in Section 4.4, mapping improves the video text recognizer accuracy and therefore we use the mapped and reordered video options for the purpose of neighbor selection. For the number of neighbors parameter $[n_B/t_B, n_A/t_A]$, we select n_B neighbors that have the highest top video match score from among t_B neighbors before the character under consideration and similarly, select n_A neighbors

from among t_A neighbors after the character.

Mapping + Relative Thresholding. Another technique, comparable to the ambiguity detection technique, uses mapping along with relative thresholding to select the neighbors for synchronization. Neighbors that have the highest ratio of their top video match score to their second highest video match score are selected. This is the technique that we have used in the experiments reported in Section 5.5.4.

Character-Specific Thresholding. As mentioned in Section 4.2.2.2, thresholding using character-specific thresholds is quite similar to the mapping operation. Therefore, as an alternative to selecting the neighbors based on the highest top video (mapped) match scores, one could select the neighbors with the highest ratio of the top video (unmapped) match score and the corresponding character-specific threshold. One should note that the proposed neighbor selection techniques do not alter the video match scores associated with the video options.

5.4.5.2 Audio Based Neighbor Selection

Audio based neighbor selection focuses on selecting neighbors that result in a lesser number of false positives when we search for their corresponding audio search term in the audio component. This is due to the fact that those audio search terms that have too many false positives may not be very reliable neighbors as they may be found within the neighbor audio time window $[W_B, W_A]$ of several audio options even when they were not actually spoken.

In the experiments reported in Table 18, we have shown that the synchronization accuracy of characters that have a larger number of phonemes in the corresponding audio search term is higher than those with a lesser number of phonemes in their audio search term. Audio search terms with larger number of phonemes lead to lesser false positives and therefore result in higher audio text recognizer accuracy. We use this observation to perform audio based neighbor selection. For the number of

neighbors parameter $[n_B/t_B, n_A/t_A]$, from among t_B neighbors before the character under consideration, we select n_B neighbors that have a higher number of phonemes in the audio search term corresponding to their top video option and n_A neighbors from t_A neighbors after the character.

5.4.6 Feature Rank Sum Based Technique

We assign feature ranks $R_1(\mathbf{a}_{j,k}(s)), R_2(\mathbf{a}_{j,k}(s)), R_3(\mathbf{a}_{j,k}(s))$ to each audio option $\mathbf{a}_{j,k}(s) \in Q(A_j(s))$ based on the ranks of the audio features F_1, F_2, F_3 among the audio options in $Q(A_j(s))$. Similar to the shorter notation for audio features, in the remainder of this chapter, we will use a shorter notation of the form R_1, R_2 and R_3 for ranks $R_1(\mathbf{a}_{j,k}(s))$, $R_2(\mathbf{a}_{j,k}(s))$ and $R_3(\mathbf{a}_{j,k}(s))$, respectively. As for rank assignments, a value of $R_1 = 1$ for an audio option indicates that the corresponding feature F_1 has the maximum value in the pruned set. Similarly, the feature ranks R_2 and R_3 are assigned to the audio options in $Q(A_j(s))$ such that the audio option with the minimum F_2 value is assigned $R_2 = 1$ and one that has maximum F_3 is assigned $R_3 = 1$. After the ranks have been assigned, the audio option with the minimum value of the rank sum (*i.e.* $R_1 + R_2 + R_3$) is chosen as the output. The index k' for the chosen audio option can be determined as follows:

$$k' = \arg\min_{k}(R_1(\mathbf{a}_{j,k}(s)) + R_2(\mathbf{a}_{j,k}(s)) + R_3(\mathbf{a}_{j,k}(s))) \tag{45}$$

where $\mathbf{a}_{j,k}(s) \in Q(A_j(s))$. In case of a tie, R_2 and R_3 values are used for tie-breaking.

Another implementation of the feature rank sum based synchronization technique makes use of the neighbor selection based audio feature $F4$ instead of the audio feature F_3 that is computed without neighbor selection. The feature rank $R_4(\mathbf{a}_{j,k}(s))$ is based on the rank of the audio feature F_4 among the audio options in $Q(A_j(s))$ such that the audio option with the highest value of F_4 is assigned R_4=1. Again, the audio option with the lowest sum of the ranks of audio features F_1, F_2 and F_4 is

selected as the synchronization output, with R_2 and R_3 being used for tie-breaking. The index k' for the selected audio option can be determined as shown below:

$$k' = \arg\min_{k}(R_1(\mathbf{a}_{j,k}(s)) + R_2(\mathbf{a}_{j,k}(s)) + R_4(\mathbf{a}_{j,k}(s))) \tag{46}$$

where $\mathbf{a}_{j,k}(s) \in Q(A_j(s))$.

5.5 Experiments

We ran a set of experiments to evaluate the effectiveness of the proposed A/V synchronization techniques. First, we ran a set of experiments to examine the effect of various factors (*i.e.* video timestamping, A/V recording alignment, VTR accuracy and ATR accuracy) on the effectiveness of the proposed A/V synchronization techniques. Then, we present a consolidated comparison between the distance based, neighbor based and feature rank sum based A/V synchronization techniques presented in this chapter. The comparison also evaluates the impact of various permutations of video timestamping accuracy and A/V recording alignment on the effectiveness of the proposed A/V synchronization techniques. Finally, we present results from our evaluation of the selective neighbor based A/V synchronization techniques.

5.5.1 Setup

The experiments were conducted using the test data set `DS-TEST-1` and another smaller test data set `DS-TEST-RA` with videos having poor A/V recording alignment. The reason for utilizing a separate smaller data set is the fact that the set `DS-TEST-1` does not contain too many characters with poor A/V recording alignment. The test data set `DS-TEST-RA` consisted of 10 videos and was used to evaluate the effectiveness of our synchronization techniques for videos with poor A/V recording alignment. The test data set `DS-TEST-RA` was recorded such that the absolute difference between the time when the character was handwritten and the time when the character was spoken was greater than 4 seconds, *i.e.* $|TS_V^w(s) - TS_A^m(s)| > 4$. For the purpose of evaluation

Table 15: Effect of video timestamping on the A/V synchronization accuracy

A/V Synchronization Technique	A/V Synchronization Accuracy (%)	
	For Poor Video Timestamping	For Good Video Timestamping
A/V Time Difference Based	36.1	85.1
A/V Neighbor Based	50.5	81.1
Feature Rank Sum Based	47.5	85.1

Setup - Mapping: `No`, Thresholding: `No (all AVC)`, Option Selection: `No (all options)`, A/V Recording Alignment: `Good`, VTR Accuracy: `Poor+Good`, ATR Accuracy: `Poor+Good`, Parameters: $[P_B, P_A]$=`[-6,16] seconds,` $[N_B, N_A]$=`[2,2],` $[W_B, W_A]$=`[-8,8] seconds`.
Poor Video Timestamping: $\delta T(s) >$ `2 seconds`
Good Video Timestamping: $\delta T(s) \leq$ `2 seconds`

the test data sets were manually labeled to contain two timestamps for each character - the manual video timestamp, $TS_V^m(s)$ and the manual audio timestamp, $TS_A^m(s)$. The A/V synchronization techniques were evaluated for all the characters in a given test data set, and ambiguity detection and option selection techniques were not used during the experiments. In the following experiments, we used $\tau_{sync} = 2$ to determine the synchronization accuracy using Equations 33 and 34.

5.5.2 Effect of the Factors that Impact A/V Synchronization

In this section, we present the results from our experiments that examine the effect of video timestamping, A/V recording alignment, VTR accuracy and ATR accuracy on the effectiveness of the proposed A/V synchronization techniques. The following experiments made use of both the test data sets `DS-TEST-1` and `DS-TEST-RA`.

Video Timestamping. To examine the effect of video timestamping on the synchronization accuracy we divided the test data sets into two categories based on the video timestamping difference $\delta T(s) = \mid TS_V^a(s) - TS_V^m(s) \mid$. Characters with $\delta T(s) > 2$ seconds were assigned to the poor video timestamping category while the characters with $\delta T(s) \leq 2$ seconds were assigned to the good video timestamping category. Table 15 shows the A/V synchronization accuracy values for the two categories

for various synchronization techniques. We observed that although the A/V synchronization accuracy for all the techniques deteriorates for the subset of characters with poor video timestamping *i.e.* for high values of $\delta T(s)$, time difference based technique suffered the maximum deterioration in A/V synchronization accuracy. This can be attributed to the fact that the time difference based technique relies solely on the difference between the automatic video timestamp $TS_V^a(s)$ and the automatic audio timestamp $TS_A^a(\mathbf{a}_{j,k}(s))$ of the audio options. In scenarios where $\delta T(s)$ is high, we may select (*i.e.* synchronize) an incorrect audio option that is closer to $TS_V^a(s)$ as opposed to the correct audio option that is closer to $TS_V^m(s)$. The deterioration in the A/V synchronization accuracy, compared to the time difference based technique, was lesser for the feature rank sum based synchronization technique as we made use of neighbors along with the A/V time difference. The effect of poor timestamping was the least on the neighbor based technique since the technique does not rely on the A/V time difference. Poor video timestamping also affects neighbors, especially if the value of $\delta T(s)$ of neighboring characters is significantly different, but this effect is less pronounced compared to the effect of A/V time difference on the accuracy of the time difference based synchronization technique. Further details of the setup of the system are provided at the bottom of Table 15.

A/V Recording Alignment. To examine the effect of A/V recording alignment on the synchronization accuracy we divided the test data sets into two categories based on the difference $\delta R(s) =\mid TS_V^m(s) - TS_A^m(s) \mid$. Characters with $\delta R(s) > 4$ seconds were considered to have poor A/V recording alignment while the characters with $\delta R(s) \leq 4$ seconds were considered to have a good A/V recording alignment. For each category, only characters with good video timestamping (*i.e.* $\delta T \leq 2$ were considered for this experiment. Table 16 shows the A/V synchronization accuracy values for the two categories for various synchronization techniques. One parameter that was significantly different compared to the previous experiment was the value

Table 16: Effect of A/V recording alignment on the A/V synchronization accuracy

A/V Synchronization Technique	**A/V Synchronization Accuracy (%)**	
	For Poor A/V Recording Alignment	For Good A/V Recording Alignment
A/V Time Difference Based	12.9	85.1
A/V Neighbor Based	44.5	78.4
Feature Rank Sum Based	41.9	82.4

Setup - Mapping: `No`, Thresholding: `No (all AVC)`, Option Selection: `No (all options)`, Video Timestamping: `Good`, VTR Accuracy: `Poor+Good`, ATR Accuracy: `Poor+Good`, Parameters: $[P_B, P_A]$=`[-32,14] seconds,` $[N_B, N_A]$=`[2,2],` $[W_B, W_A]$=`[-8,8] seconds`.
Poor A/V Recording Alignment: $\delta R(s) >$ `4 seconds`
Good A/V Recording Alignment: $\delta R(s) \leq$ `4 seconds`

of parameter $[P_B, P_A]$. To accommodate for the significant skew between the writing and the utterance of a character by the instructor, *i.e.* poor A/V recording alignment, we used a larger $[P_B, P_A]$ window. As a result of the large values for $[P_B, P_A]$, the synchronization techniques are presented with a large number of candidate audio options. For the subset of characters with good A/V recording alignment, as expected, the time difference based synchronization technique outperformed other techniques. However, for the subset with poor A/V recording alignment the time difference based synchronization suffered a very significant degradation in performance due to very high values of δR. For poor A/V recording alignment, due to lesser or no reliance on A/V time difference based feature, the degradation in synchronization accuracy was lesser for the feature rank sum based and the neighbor based techniques. More details about the experimental setup are provided below the table of results.

VTR Accuracy. While VTR accuracy does not directly impact the A/V synchronization of a character, but it affects the reliability of video options that are used as neighbors for calculation of synchronization feature F_3. The following experiment was conducted with a total of 4 neighbors, 2 before the character under consideration and 2 after the character under consideration. To demonstrate the impact of the VTR

Table 17: Effect of the VTR accuracy of the neighbors on the A/V synchronization accuracy

A/V Synchronization Technique	**A/V Synchronization Accuracy (%)**	
	For Poor VTR Accuracy of Neighbors	For Good VTR Accuracy of Neighbors
A/V Neighbor Based	37.2	65.1
Feature Rank Sum Based	35.7	58.9

Setup - Mapping: `No`, Thresholding: `No (all AVC)`, Option Selection: `No (all options)`, Video Timestamping: `Good`, A/V Recording Alignment: `Poor`, ATR Accuracy: `Poor+Good`, Parameters: $[P_B, P_A]$=`[-32,14] seconds,` $[N_B, N_A]$=`[2,2],` $[W_B, W_A]$=`[-8,8] seconds`.
Poor VTR Accuracy: $\leq$ `50% of neighbors are correctly recognized by VTR`
Good VTR Accuracy: $>$ `50% of neighbors are correctly recognized by VTR`

accuracy of the neighbors on A/V synchronization the characters in the test data set were divided into two categories, one with poor VTR accuracy for the neighbors and the other with good VTR accuracy for the neighbors. Here, poor VTR accuracy for the neighbors refers to the characters that have up to 50% of the neighbors correctly recognized, *i.e.* 0-2 out of the 4 neighbors have the correct output as their top video option. The good VTR accuracy scenario refers to those characters that have more than 50% (*i.e* 3-4 out of the 4 neighbors) of the neighbors correctly recognized. Table 17 shows the effect of the VTR accuracy of the neighbors on the A/V synchronization accuracy. We observed that the A/V synchronization accuracy, for each of the synchronization techniques that made use of neighbors, deteriorates when the VTR accuracy of the neighbors was low. This can be attributed to the fact that in presence of incorrectly recognized neighbors, synchronization techniques attempt to locate utterances of the wrong characters in the vicinity of the audio utterance of the character under consideration. Therefore the value of the audio feature F_3 may not be the maximum for the correct audio option. Further details of the setup of the system are provided at the bottom of Table 17.

ATR Accuracy. Although we cannot compute the ATR accuracy for our system,

Table 18: Effect of the ATR accuracy on the A/V synchronization accuracy

A/V Synchronization Technique	**A/V Synchronization Accuracy** (%)	
	For Poor ATR Accuracy	For Good ATR Accuracy
A/V Time Difference Based	25.9	50.5
A/V Neighbor Based	45.5	61.1
Feature Rank Sum Based	41.2	57.8

Setup - Mapping: `No`, Thresholding: `No (all AVC)`, Option Selection: `No (all options)`, Video Timestamping: `Poor`, A/V Recording Alignment: `Good`, VTR Accuracy: `Poor+Good`, Parameters: $[P_B, P_A]$=`[-6,16] seconds,` $[N_B, N_A]$=`[2,2],` $[W_B, W_A]$=`[-8,8] seconds`.
Poor ATR Accuracy: `1-2 phonemes in search string`
Good ATR Accuracy: `3+ phonemes in search string`

we have observed that the ATR accuracy is higher for characters that have a higher number of phonemes in their audio equivalents or search terms, such as `'seven'` and `'plus'`, as opposed to characters that have a smaller number of phonemes in their search terms, such as `'b'`,`'c'`,`'d'` and `'e'`. Therefore, we made use of the number of phonemes as an approximate measure of the ATR accuracy for a given character. Similar to the previous experiments, the characters in the test data sets were divided into two categories, one with poor ATR accuracy and the other with good ATR accuracy. The set with poor ATR accuracy consisted of the characters that had up to 2 phonemes in their search term and the set with good ATR accuracy consisted of the characters that had 3 or more phonemes in their search term. Each of the proposed synchronization techniques was evaluated for the two sets of characters. Table 18 shows the effect of the ATR accuracy on the A/V synchronization accuracy. As expected, for each of the proposed synchronization techniques, we observed an increase in the A/V synchronization accuracy for the set of characters with good ATR accuracy when compared to the set with poor ATR accuracy. Further details about the experimental setup are provided at the bottom of Table 18.

Table 19: Choosing the suitable A/V synchronization technique and parameter values

S.No.	Parameter Values			A/V Synchronization Accuracy (%)		
	Pruning Time Window $[P_B, P_A]$ (sec)	Number of Neighbors $[N_B, N_A]$	Neighbor Time Window $[W_B, W_A]$ (sec)	Good TS Good RA	Poor TS Good RA	Good TS Poor RA
Using the A/V Time Difference Based A/V Synchronization Technique						
1a.	[-1,1]	-	-	64.9	4.9	0.0
1b.	[-6,8]	-	-	85.1	33.4	5.3
1c.	[-6,16]	-	-	85.1	36.1	10.2
1d.	[-32,14]	-	-	85.1	36.1	25.9
Using the A/V Neighbor Based A/V Synchronization Technique						
2a.	[-6,8]	[2,2]	[-8,8]	83.6	44.4	14.4
2b.	[-6,16]	[2,2]	[-8,8]	81.1	50.5	22.3
2c.	[-32,14]	[1,1]	[-8,8]	79.7	41.8	33.5
2d.	[-32,14]	[2,2]	[-8,8]	78.4	48.3	47.6
2e.	[-32,14]	[4,4]	[-8,8]	76.2	45.2	45.4
2f.	[-50,30]	[2,2]	[-8,8]	74.3	44.8	43.6
Using the Feature Rank Sum Based A/V Synchronization Technique						
3a.	[-6,8]	[2,2]	[-8,8]	86.5	41.0	12.6
3b.	[-6,16]	[2,2]	[-8,8]	85.1	47.5	20.3
3c.	[-32,14]	[2,2]	[-8,8]	82.4	45.1	41.9

Setup - Mapping: `No`, Thresholding: `No (all AVC)`, Option Selection: `No (all options)`, VTR Accuracy: `Poor+Good`, ATR Accuracy: `Poor+Good`.
Good TS: $\delta T(s) \leq$ `2 seconds`, Poor TS: $\delta T(s) >$ `2 seconds`
Good RA: $\delta R(s) \leq$ `4 seconds`, Poor RA: $\delta R(s) >$ `4 seconds`

5.5.3 Choosing the A/V Synchronization Technique and the Parameter Values

The aim of the following set of experiments was to (1) for a given A/V synchronization technique, understand the impact of the variation in the value of the pruning

window $[P_B, P_A]$ and the number of neighbors window $[N_B, N_A]$ on the synchronization accuracy, and (2) gain an understanding, which based on the video timestamping and recording alignment characteristics associated with a given set of videos help us choose the most suitable A/V synchronization technique (from among the time difference based, the neighbor based and the feature rank sum based synchronization technique). The consolidated results from the experiments are reported in Table 19 and are explained below. Again, all the videos from the test data sets `DS-TEST-1` and `DS-TEST-RA` were used for the experiments.

Pruning Time Window $[\mathbf{P_B}, \mathbf{P_A}]$**.** The pruning time window $[P_B, P_A]$ is used to limit the set of audio options that are considered as possible candidates for A/V synchronization. The first parameter of the pruning window *i.e.* P_B is meant to accommodate the skew (between the appearance of handwritten content and the corresponding audio utterance) that occurs due to A/V recording alignment ($\delta R(s)$) and the second parameter *i.e.* P_A is meant to accommodate the skew that occurs due to video timestamping ($\delta T(s)$) and occlusions. In the test data sets, the skew due to video timestamping was less than 10 seconds, while that caused by A/V recording alignment was less than 30 seconds. Any skew caused by occlusions was no more than 10 seconds. For the good TS (timestamping) and good RA (recording alignment) case, a pruning time window $[P_B, P_A]$ = [-6,8] was sufficient as the P_B value accommodates a maximum $\delta R(s) = 4$ seconds with a margin of 2 seconds while the P_A accommodates a maximum occlusion of $\delta R(s) = 4$ seconds and a $\delta T(s) = 2$ seconds and a margin of 2 seconds. Similarly, for the poor TS and good RA scenario, the P_A was increased to accommodate a maximum $\delta T(s) = 10$ seconds. For the poor RA and good TS scenario, the P_B was increased to accommodate a maximum $\delta R(s)$ = 30 seconds with a 2 second margin and the P_A was increased to accommodate a maximum occlusion of 10 seconds, $\delta T(s) = 2$ seconds and a 2 second margin.

As expected, each category (based on TS and RA) of characters attains the best

synchronization results when used with the pruning time window $[P_B, P_A]$ that was designed for the specific category. For instance, the A/V synchronization accuracy for characters with good TS and good RA was maximum (85.1%) for the pruning time window $[P_B, P_A]$ = [-6,8] that was designed for it. For good TS and good RA, any further increase in $[P_B, P_A]$ to [-6,16] or [-32,14] (rows 1c-d) did not lead to an increase in the synchronization accuracy as the closest audio option (one that lies within the window [-6,8]) continues to be the one selected as the synchronized audio option. For poor TS and good RA, we observed the maximum synchronization accuracy for $[P_B, P_A]$ = [-6,16] (row 1c) and there was no further increase when this window was further expanded (row 1d). Finally, for the good TS and poor RA scenario, we observed best synchronization results for the corresponding pruning time window $[P_B, P_A]$ = [-32,14] (row 1d). The table also shows that using a smaller than required pruning time window $[P_B, P_A]$ may degrade the synchronization accuracy as the correct audio option may not be present within this pruning time window. This can be observed in row 1a for good TS and good RA, rows 1a-b for poor TS and good RA and row 1a-c for good TS and poor RA.

Number of Neighbors $[\mathbf{N_B}, \mathbf{N_A}]$***.*** The effect of changing the number of neighbors $[N_B, N_A]$ that are used for synchronization on the corresponding A/V synchronization accuracy is shown in rows 2c-e. For our implementation of the recognition system, $[N_B, N_A] = [2, 2]$ was found to correspond to the maximum synchronization accuracy. We observed that when the number of neighbors was reduced to $[1, 1]$ (row 2c), the synchronization accuracy decreased for the poor TS and poor RA scenarios and increased for the good TS and good RA scenario. This can be attributed to the fact that the video neighbors are not always correctly recognized by the VTR, using just 1 neighbor on each side may be insufficient for synchronization. In several cases, audio options end up with only one video neighbor in their audio neighborhood (*i.e.* within the neighbor time window $[W_B, W_A]$ = [-8,8]) and result in a tie. As explained

in Section 5.4.4, we use the audio feature F_2 for tie-breaking, which is the same as the output of the time difference based technique. This causes the synchronization accuracy to increase for the good TS and good RA scenario (row 2c) to a value comparable to the time difference technique based (row 1d). Similarly, in the poor TS and poor RA scenarios, the synchronization accuracy (row 2c) degraded to a value comparable to the corresponding value for the time difference based technique (row 1d). We also observed that using too many neighbors *i.e.* $[N_B, N_A]$ = [4,4], may also lead to a slight degradation in the synchronization accuracy. This can be attributed to the size of the pruning time window used and the low VTR accuracy,

Neighbor Audio Time Window $[\mathbf{W_B}, \mathbf{W_A}]$***.*** The neighbor audio time window $[W_B, W_A]$ should be chosen based on the number of neighbors $[N_B, N_A]$ that are considered for synchronization and also the average time between the utterance of consecutive characters. Although not shown in Table 19, we have observed both experimentally and also by inspecting the test data set that for $[N_B, N_A]$ = [2,2], a neighbor audio time window of $[W_B, W_A]$ = [-8,8] seconds results in the best synchronization accuracy. As expected, if $[W_B, W_A]$ is too small, the neighbors may lie outside the window and if $[W_B, W_A]$ is too large, audio occurrences (false positives) that do not correspond to the neighbor may appear in the window. Both these scenarios will lead to an incorrect value for the feature F_3 and a degradation in the performance of any neighbor based technique. For the remainder of the experiments, we use a neighbor audio time window of $[W_B, W_A]$ = [-8,8] when $[N_B, N_A]$ = [2,2].

Time Difference Based Technique. As expected, for the time difference based technique (row 1b) the maximum synchronization accuracy was achieved for good TS and good RA. Since the time difference based technique selects the audio option based entirely on the time difference between the video option and the candidate audio option, the synchronization accuracy was significantly lower for characters with poor TS. The synchronization accuracy degraded even further for characters with poor

RA and even with a suitable pruning time window $[P_B, P_A] = [-32, 14]$ (row 1d) the maximum synchronization accuracy for characters with poor RA is not much better. The A/V time difference based technique does not make use of neighbors and therefore the parameters $[N_B, N_A]$ and $[W_B, W_A]$ are not relevant here.

Neighbor Based Technique. Using neighbors for synchronization helps in improving the synchronization accuracy for characters with poor TS or poor RA. From the set of experiments reported in Table 19, we can see that the use of A/V neighbor based synchronization (row 2b) resulted in an improvement in the synchronization accuracy when compared to the time difference based technique (row 1c), for the same pruning time window of $[P_B, P_A]$ = [-6,16] for the poor TS scenario. Similarly, for the poor RA scenario, we saw a significant increase in the synchronization accuracy (row 2d) when using the A/V neighbor based synchronization compared to the time difference based technique (row 1d), for the pruning time window $[P_B, P_A]$ = [-32,14]. However, by using neighbor based synchronization, the synchronization accuracy deteriorates for the good TS and good RA scenario (in row 2a) when compared to the time difference based technique (row 1b) since the neighbors may not be very reliable and lead to more incorrect synchronization when compared to the time difference based technique which works extremely well for this scenario. The results also demonstrate that increasing the $[P_B, P_A]$ further to a value of [-50,30] (row 2f) could degrade the synchronization accuracy. Increasing the pruning time window beyond the required size leads to more audio options being considered as candidates and due to factors like low VTR and low ATR accuracy this may lead to an incorrect audio option being assigned a higher value for feature F_3. This impacts the accuracy of synchronization techniques that make use of feature F_3. The maximum A/V synchronization accuracy achieved by the A/V neighbor based technique for the poor TS and good RA scenario was 50.5% (row 2b) and for the good TS and poor RA scenario was 47.6% (row 2d).

Feature Rank Sum Based Technique. Feature rank sum combines the advantages of the time difference based technique, the neighbor based technique and incorporates the audio match score as a feature in the synchronization process. Experiments show that the feature rank sum based technique is a good compromise between the time difference based and the neighbor based techniques. For the good TS and good RA scenario, this technique performed better (row 3a) than the time difference technique (row 1b) for the pruning time window $[P_B, P_A]$ = [-6,16], due to the advantage of using the audio match score values (audio feature F_1) and did not degrade as a result of using the neighbors since the small pruning time window that was used for this experiment did not allow many other audio options with a high number for audio feature F_3. For the poor TS and poor RA scenario, the maximum synchronization accuracies achieved using feature rank sum (row 3b for poor TS and row 3c for poor RA) were higher than those of the time difference technique (rows 1c-d) but lower than those of the neighbor based technique due to the effect of A/V time difference audio feature F_2. The feature rank sum technique that worked reasonably well for all three categories of TS and RA is shown in row 3c and the synchronization accuracies for the three categories were 82.4%, 45.1% and 41.9%.

One should note that in a given test data set, there may be characters that fall into each of the three categories based on the A/V recording alignment of the videos as well as the quality of the timestamping algorithm. The ratio of characters that fall into each of these three categories determine the choice of A/V synchronization technique. The feature rank sum based technique, as shown above, is a reasonable choice for most of the test data sets.

5.5.4 Effect of neighbor selection on the A/V synchronization accuracy

The following set of experiments was conducted to examine the effect of neighbor selection on the neighbor based and the feature rank sum based synchronization

Table 20: Effect of neighbor selection on the A/V synchronization accuracy

S.No.	A/V Synchronization Technique	A/V Synchronization Accuracy (%)		
		Good TS Good RA	Poor TS Good RA	Good TS Poor RA
1.	A/V Time Difference	85.1	36.1	25.9
2a.	A/V Neighbor♣	78.4	48.3	47.6
2b.	Selected Neighbor (Video)♠	83.0	56.4	54.7
2c.	Selected Neighbor (Audio)♠	80.3	51.2	50.0
3a.	Feature Rank Sum♣	82.4	45.1	41.9
3b.	Feature Rank Sum (Video))♠	87.0	50.9	47.8
3c.	Feature Rank Sum (Audio)♠	84.9	48.3	46.1

Setup - Thresholding: `No (all AVC)`, Option Selection: `No (all options)`,
Parameters: $[P_B, P_A]$ = [-32,14], $[N_B, N_A]$ = [2,2]♣, $[n_b/t_b, n_a/t_a]$ = [2/3,2/3]♠, $[W_B, W_A]$ = [-8,8].
Good TS: $\delta T(s) \leq$ `2 seconds`, Poor TS: $\delta T(s) >$ `2 seconds`
Good RA: $\delta R(s) \leq$ `4 seconds`, Poor RA: $\delta R(s) >$ `4 seconds`

techniques. In Table 20, rows 2a and 3a show the results from neighbor based and the corresponding feature rank sum based synchronization techniques without the use of neighbor selection. We used $[N_B, N_A]$ = [2,2] for these experiments. Rows 2b and 3b show the corresponding results for the same setup, except for the fact that we made use of video based neighbor selection with $[n_b/t_b, n_a/t_a]$ = [2/3,2/3], *i.e.* 2 out of 3 neighbors were selected from the neighbors on each side of the video option being synchronized. Neighbors with a higher value of the ratio of their top video option's score and the second video option's score were selected for synchronization. The experiments were setup to use mapping for video match scores. The results with the use of audio based neighbor selection are shown in rows 2c and 3c for $[N_B, N_A]$ = [2/3,2/3]. The neighbors with a larger number of phonemes in the audio search term were chosen for synchronization.

For various categories of characters, based on TS and RA, we observed that the use of neighbor selection improves the synchronization accuracy of both the neighbor

based technique and the feature rank sum based technique. We also observed that video based neighbor selection outperforms audio based neighbor selection for both the neighbor based technique and the feature rank sum based technique. For a given test data set, if the majority of the characters have either poor TS or poor RA, then the neighbor based technique with video based neighbor selection (row 2b) would be the synchronization technique of choice. If the majority of the characters have good TS and good RA, then the feature rank sum based technique that makes use of the video based neighbor selection (row 3b) would be the technique of choice. This would also be a good option for techniques that have characters well distributed between the three categories.

5.6 Summary

In this chapter we proposed a set of A/V synchronization techniques that could be used to synchronize the output of the video and the audio text recognizers. Towards this end, the proposed techniques, for each selected video option of a given ambiguous character, attempt to locate the audio option that corresponds to the most likely utterance of the video option in the audio component of the video. Among the proposed A/V synchronization techniques, the time difference and neighbor based techniques are geared towards videos with a specific accuracy of the video timestamping and a specific quality of recording alignment, whereas the feature rank sum based technique, also proposed in this chapter, is suitable for videos whose timestamping accuracy and recording alignment quality values maybe unknown or spread across a range. To further improve the accuracy of the neighbor based and the feature rank sum based technique, we also proposed a method for selecting the neighbors, from the vicinity of the video option being synchronized, that are used for synchronization. The selection is based on the expected video or audio text recognition accuracy of the possible neighbors and leads to more accurate values for the neighbor feature that is

used by the neighbor based and the feature rank sum based technique. Experiments showed that the proposed synchronization techniques were able to achieve high accuracy values for videos with good video timestamping and good recording alignment. Even when the video timestamping accuracy was not very high or the recording alignment was significantly skewed the proposed synchronization techniques were able to achieve significantly better synchronization accuracy values compared to the simple time difference based synchronization technique.

CHAPTER VI

AUDIO-VIDEO COMBINATION

6.1 Introduction

The output from multiple recognizers, based on the domain, can be combined using several different methods that range from weighted combination, to rank based combination, to methods that rely on sophisticated machine learning algorithms for combination. Such combination methods, in the context of the audio-video combination system, presented in this dissertation, provide us an opportunity to incorporate various recognizer, character and instructor specific nuances, learned or otherwise, into the end-to-end recognition system. In this chapter, we describe several audio-video combination methods [113, 112] that could be used by the end-to-end recognition system.

The input to the audio-video combination stage is a set of segmented characters, each with a set of corresponding video options that are in turn associated with at most one audio option each. The audio-video combination stage generates the final recognized output based on the match scores generated by the two recognizers, the computed audio-video features and may also use some parameters derived from running the recognizers on the training data. In the past, similar combination based approaches have been used in several related contexts, for instance, to improve the recognition accuracy of handwriting recognizers [120], speech recognizers [25] and a combination of the two [45]. We have evaluated several rank-level and measurement-level combination techniques [109] such as rank sum and weighted sum rule using classifier-specific weights and character-specific weights. We have also used an ensemble of audio-video recognizers, each configured with different parameters and

combination techniques, for improving the character recognition accuracy.

The remainder of this chapter is organized as follows. Section 6.2 provides a recap of the mathematical model associated with the audio-video combination component. The combination techniques and associated examples are described in Section 6.3. Section 6.4 presents a detailed evaluation of the proposed combination techniques. Finally, a summary of the contributions is presented in Section 6.5.

6.2 Audio-Video Combination - Mathematical Model

In this section, we revisit the mathematical model associated with the audio-video (A/V) combination component of our end-to-end recognition system. The combination component receives its input in the form of synchronized video and audio options from the synchronization component. The input consists of the set of ambiguous segmented characters S_D, where each segmented character $s \in S_D$ is associated with a set of video options $O(s)$ and each video option $\mathbf{v}_j(s) \in O(s)$ is further associated with at most one synchronized audio option $\mathbf{a}_{j,k'}(s)$. When a segmented character along with the set of synchronized video and audio is passed to the combination component, a single video option $\mathbf{v}_{j'}(s)$ from amongst the set of input synchronized video and audio options is produced as the output, and we represent this process by the function Z,

$$Z(s) = \mathbf{v}_{j'}(s) \tag{47}$$

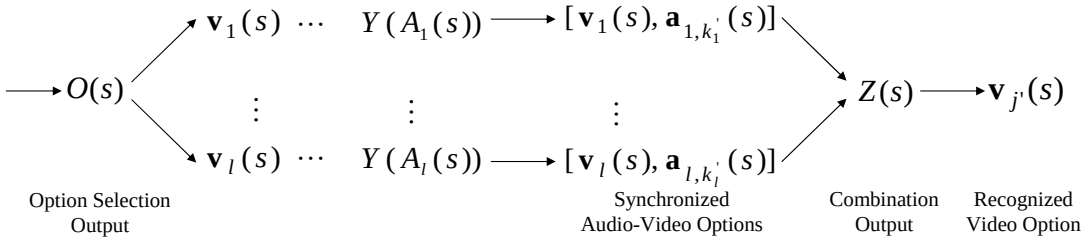


Figure 24: Output of option selection, audio-video synchronization and subsequent audio-video combination for generation of the recognized video option

Figure 24 shows, for a given ambiguous segmented character $s \in S_D$, the transformation from the set of selected video options $O(s)$ to the recognized video option $\mathbf{v}_{j'}(s)$. The recognition accuracy for the combination component over the set of ambiguous segmented characters can be determined using the following equation.

$$\alpha_Z(S_D) = \frac{\sum_{\forall s \in S_D} E(v^c_{j'}(s), G(s))}{|S_D|} \tag{48}$$

where the function E, defined in Equation 3, determines the equality of the two supplied parameters and produces 1 if the parameters are equal and 0 otherwise. The end-to-end character recognition accuracy, for the set of segmented characters S, can therefore be calculated as follows

$$\alpha(S) = \frac{|S - S_D| \times \alpha_V(S - S_D) + |S_D| \times \alpha_Z(S_D)}{|S|} \tag{49}$$

where $\alpha_V(S - S_D)$ is the accuracy of the video text recognizer for the set of non-ambiguous segmented characters.

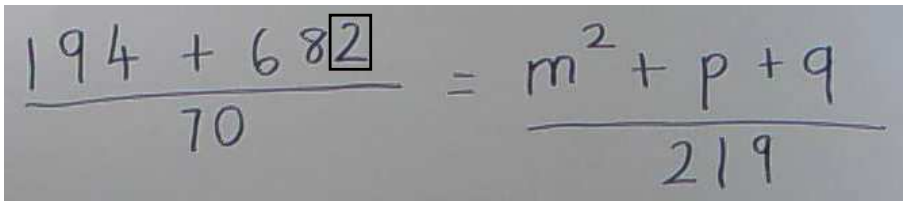
6.3 Combination Techniques

This section describes the specifics of the combination techniques that have been implemented into the audio-video combination component. Where necessary, we have used real examples to demonstrate the working of the combination techniques.

6.3.1 Rank Based Techniques

A challenge often encountered by classifier combination techniques results from the fact that the confidence scores generated by the different classifiers are not normalized with respect to each other. In such situations, a commonly used combination technique is the rank sum (Borda count [109]) technique. The rank sum technique, based on the confidence scores (or feature values), assigns ranks to the various options (or features) generated by each classifier. For a given recognition output, its

Character under consideration:



Working of the Rank Sum Based Technique:

Video Option	Audio Option	Video Match Score & Rank		Audio Match Score & Rank		Rank Sum	Decision
v_j^c	$a_{j,k'}^c$	v_j^p	R_v	$a_{j,k'}^p$	R_a	$R_{sum} = R_v + R_a$	$= \min(R_{sum})$
Z	Z	0.98	1	0.23	4	5	
2	2	0.89	2	0.65	2	4	4
a	a	0.82	3	0.51	3	6	
l	l	0.81	4	0.14	5	9	
7	7	0.81	5	0.77	1	6	

Character name (Truth) : 2
Character name (Recognized) : 2

Figure 25: Example demonstrating the working of the rank sum based technique

ranks are summed across all the classifiers and the recognition output with the lowest rank is chosen as the output. The rank sum technique is often accompanied by a suitable tie-breaking strategy. As is evident, one of the advantages of using a rank based technique here is that there is no need to compute weights for the audio and video recognizers or normalize the match scores. We have implemented rank sum for different subsets of the video match score, audio match score and the audio features. For the purpose of tie-breaking we make use of the audio feature $F_2(\mathbf{a}_{j,k'}(s))$ (defined in Chapter 5 as the absolute time difference between the timestamp of the video and the audio option being combined) and select the option with lower corresponding value of the audio feature F_2.

Figure 25 shows an example demonstrating the working of the rank sum based combination technique. The example makes use of ranks derived from the video and

the audio match scores to combine the output of the two recognizers. Note that in absence of rank sum based method, a method that would have relied on simple summation of the match scores would have selected the character ‘7’ as the output. The rank sum based combination method is able to eliminate the error in the output of the video text recognizer and is able to select the correct output. The significantly high audio match score for the audio option corresponding to the character ‘a’ can be attributed to the presence of the phoneme for character ‘a’ in the phoneme sequence for the character ‘8’, which occurs in the vicinity. The same holds true for the character ’7’, where the actual character appears in the vicinity.

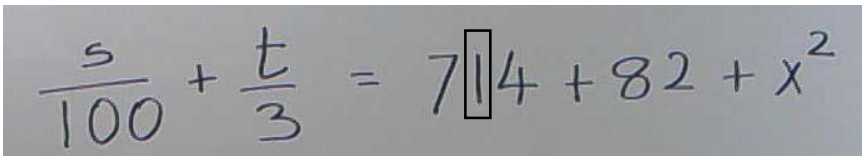
6.3.2 Recognizer-Specific Weight Based Techniques

While the sum rule based technique, to some degree, helps us eliminate the issues related to video and audio match score normalization, it does not incorporate into its framework the fact that different recognizers may have different accuracy values. Furthermore, in the case of audio-video based mathematical content recognition from classroom videos the accuracy values for each recognizer may be different for different instructors. The recognizer-specific weight based technique [109] helps us incorporate such recognizer accuracy value specific considerations into the determination of the final output. The video text recognizer is assigned a weight w_V and the audio text recognizer is assigned a weight w_A. The combination technique then uses these weights to calculate combined match score (weighted sum of video and audio match scores) and selects the video option with highest combined match score as the output. The combination function $Z(s)$, in this case, can be represented as follows

$$Z(s) = (v^c_{j'}(s) \mid j' = \underset{j:\mathbf{v}_j(s)\in O(s)}{\arg\max} (w_V \times v^p_j(s) + w_A \times a^p_{j,k'}(s))) \tag{50}$$

As a rule of thumb, the weights assigned to each recognizer should be proportional to the accuracy of the recognizer. Such weights can also be estimated using the

Character under consideration:



Working of the Recognizer-Specific Weight Based Technique

Video Option v_j^c	Audio Option $a_{j,k'}^c$	Video Match Score v_j^p	Audio Match Score $a_{j,k'}^p$	Weighted Sum $p = w_V \times v_j^p(s) + w_A \times a_{j,k'}^p(s)$	Decision $\max(p)$
l	l	0.99	0.27	0.70	
1	1	0.89	0.60	0.77	0.77
(	(	0.86	-	0.51	
I	i	0.85	0.12	0.56	
i	i	0.84	0.12	0.55	

Recognizer-specific weights $w_V = 0.6, w_A = 0.4$

Character name (Truth) : 1
Character name (Recognized) : 1

Figure 26: Example to demonstrate the working of the recognizer-specific weight based technique

training data set. Training data set that is specific to an instructor can be used to estimate the best weight values for a known instructor. In our implementation of the recognizer-specific weights based technique, although not necessary, we ensure that the sum of recognizer weights is 1 (*i.e.* $w_V + w_A = 1$).

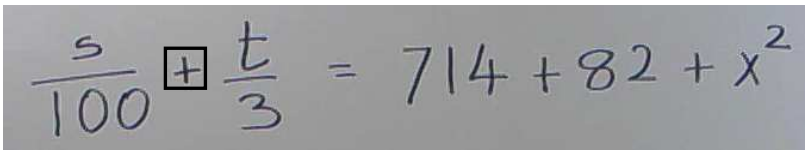
In the example shown in Figure 26, we have used $w_V = 0.6$ and $w_A = 0.4$. Although the video text recognizer's top video option is the character '`l`', upon combination with the synchronized output from the audio text recognizer, character '`1`' is chosen as the recognition output. This example again demonstrates that a combination of audio and video text recognizer can help reduce the errors in the output of the standalone video text recognizer.

6.3.3 Character-Specific Weight Based Techniques

The accuracy of the video and the audio text recognizer can be significantly different for different characters. For instance, in case of the word-spotter based audio text recognizer, characters that correspond to a longer phoneme sequence are more accurately recognized by the audio text recognizer. Similarly, based on our analysis of the output from the GOCR based video text recognizer, certain characters (*e.g.* `'m'` and `'n'`) are more accurately identified by the recognizer as opposed to other characters. The use of a different set of weights for every character, therefore, has the potential to further improve the recognition accuracy. We use $w_V(c)$ and $w_A(c)$ to denote the character-specific weights for a character $c \in C$ (note that a character $c \in C$, in some cases, may correspond to multiple audio search strings and therefore the audio weight is specific to the audio search string). It is ensured that for any given character $c \in C$, $w_V(c) + w_A(c) = 1$.

One possible way to compute the character-specific video weight $w_V(c)$ is to use the video text recognizer's accuracy-related metrics, such as precision and sensitivity, that correspond to the character $c \in C$. However, for the word-spotter based audio text recognizer, due to the way the word-spotter operates the accuracy related metrics, and therefore the audio weight $w_A(c)$, cannot be calculated directly. This problem stems from the fact that the word-spotter attempts to locate a sequence of phonemes in an audio segment and in several cases phonemes from certain characters (*e.g.* `'e'`) appear as part of other characters (*e.g.* `'equals'`), leading to a large number of false positives for characters like `'e'` that are not actually false positives. We have chosen to assign relatively higher audio weights to characters that have larger number of phonemes and lower to those with smaller number of phonemes. In our implementation, for a given character c, we assign a *fraction* of $(1 - w_V(c))$ as the audio weight and the remaining fraction is added to the video weight. The *fraction* is computed as $min(1, (\#\text{ of phonemes in } c)/M_p)$. In our implementation we have used

Character under consideration:



Working of the Character-Specific Weights Based Technique:

Video Option v_j^c	Audio Option $a_{j,k'}^c$	Video Match Score v_j^p	Audio Match Score $a_{j,k'}^p$	Char-Specific Weights: Video $w_V(c)$	Char-Specific Weights: Audio $w_A(c)$	Weighted Sum $p = w_V(c) \times v_j^p + w_A(c) \times a_{j,k'}^p$	Decision $\max(p)$
+	+	0.87	0.33	0.88	0.12	0.80	0.80
i	i	0.84	0.17	0.58	0.42	0.56	
t	t	0.80	0.64	0.60	0.40	0.74	
j	j	0.75	0.08	0.56	0.44	0.46	
f	f	0.70	0.04	0.15	0.85	0.14	

Character name (Truth) : +
Character name (Recognized) : +

Figure 27: Example demonstrating the working of the character-specific weight based technique

$M_p = 1$ and $M_p = 5$. When using $M_p = 1$, the audio weight is always $(1 - w_V(c))$. $M_p = 5$ roughly corresponds to the 90^{th} percentile in terms of number of phonemes contained in the audio search strings for all characters, and as a result higher audio weights are assigned to audio search strings with more phonemes.

$$Z(s) = (v_{j'}^c(s) \mid j' = \underset{j:\mathbf{v}_j(s) \in O(s)}{\arg\max} (w_V(v_j^c(s)) \times v_j^p(s) + w_A(a_{j,k'}^c(s)) \times a_{j,k'}^p(s))) \quad (51)$$

Figure 27 shows the working of the character-specific weights based technique. In the example, notice that the audio option corresponding to the third video option `'t'` has a significantly high value for audio match score. This high value of the audio match score can be attributed to the fact that there is a character `'t'` that immediately follows the character `'+'` (the truth). In this scenario, if one had used a simple sum rule then the character would have been incorrectly recognized as character `'t'`.

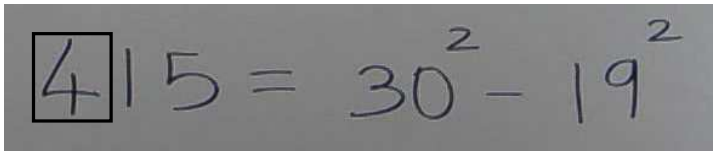
However, the character-specific weights based combination technique is able to determine the correct output, *i.e.* the character '+'. The output of the combination stage for the real character 't', although not shown in the example, was also correctly determined by the character-specific weights based combination technique.

6.3.4 Recognizer Ensemble Based Techniques

The recognizer ensemble based technique is a two-stage combination technique. As part of the first stage, the technique creates an ensemble of recognizers, where each individual recognizer uses a different combination technique (from the ones proposed above) or uses a different parameter (*e.g.* recognizer-specific weight). The implementation of each recognizer was altered such that, for each segmented character, the recognizer produces a ranked list of multiple recognition results as output. Thus, the recognition result that is ranked 1 is the final output that the recognizer would have otherwise produced. The number of ranked recognition results generated for a given segmented character is contingent on the option selection technique. In our implementation of the recognizer ensemble based technique we have limited the number of recognizers contained in a given ensemble to 4. The second stage of the recognizer ensemble based technique is responsible for combining the output generated by the recognizers. The technique utilized for combination in this stage is the rank sum based combination technique. This is similar to the technique described in Section 6.3.1, except for the fact that there may be more than 2 recognizers whose output needs to be combined. At the end of the second stage, for a given segmented character, the recognition result with the lowest rank sum is chosen as the final output.

To determine the set of recognizers that constitute the ensemble at the end of first stage, we exhaustively evaluate various combinations of the recognizers using the training data set. The ensemble that results in the highest recognition accuracy is the one used for the test data set. In cases where the number of recognizers

Character under consideration:



Working of the Recognizer Ensemble Based Technique:

1st Level of Combination

Video Option	Audio Option	Video Text Recognizer		Combination using Classifier Specific Weights [0.8,0.2]		Combination using Character Specific Weights		Rank Sum Combination	Decision
v_j^c	$a_{j,k'}^c$	v_j^p	Rank R_1	A/V Value	Rank R_2	A/V Value	Rank R_3	$R_{sum} = R_1 + R_2 + R_3$	$= \min(R_{sum})$
G	G	0.90	1	0.75	2	0.17	5	8	
h	h	0.84	2	0.75	3	0.53	4	9	
4	4	0.79	3	0.76	1	0.73	2	6	6
m	m	0.78	4	0.68	5	0.65	3	12	
+	+	0.78	5	0.69	4	0.74	1	10	

Character name (Truth) : 4
Character name (Recognized) : 4

Figure 28: Example demonstrating the working of the recognizer ensemble based technique

that can be added to the ensemble is high and exhaustive search is not possible, techniques like genetic algorithms [19] can be used to determine the set of recognizers for the ensemble. For the second stage of the recognizer ensemble based combination technique, a number of other techniques like majority voting [109], character-specific recognizer selection [89], etc. can be employed.

The example shown in Figure 28 demonstrates the working of the recognizer ensemble based audio-video combination technique. The example shows an ensemble consisting of 3 recognizers, which are then combined using the rank sum method for determining the final output. The first recognizer in the ensemble is the video text recognizer, the second is a recognizer that uses classifier-specific weight based combination method ($w_V = 0.8$, $w_A = 0.2$) and the third recognizer uses the character-specific weights based combination method (using precision for $w_V(c)$ and $1 - w_V(c)$ for $w_A(c)$). Notice that the character recognized with highest match score by the

video text recognizer (*i.e.* highest rank) is assigned the lowest rank by the third recognizer. Upon rank sum based second level combination the recognition output is the character '4', which is the correct output. The character '5' (results not shown in the example) was also correctly recognized by the above ensemble of recognizers. This is interesting because no single recognizer from the ensemble was independently able to correctly recognize both these characters (the characters '4' and '5').

6.4 Experiments

This section presents an evaluation of the proposed audio-video based combination techniques using the 2 test data sets, DS-TEST-1 and DS-TEST-2. The corresponding training data sets, DS-TRAIN-1 and DS-TRAIN-2, were also used for some of the experiments. Since the ambiguity detection and the synchronization techniques have already been extensively evaluated in the preceding chapters, in this section, we focus on evaluating and comparing the performance of various audio-video based combination techniques. In all the reported experiments, the ambiguity detection and the option selection components were configured to use the relative threshold method with a threshold value 0.9. The synchronization component was configured to use the Rank Sum based technique.

The results for the test data set DS-TEST-1 are reported in Table 21, and Table 22 reports the results for the test data set DS-TEST-2. The purpose of conducting the evaluation using 2 data sets, recorded by 2 different instructors, was to demonstrate that our techniques can be seamlessly applied to other data sets as well, as long as the recorded classroom videos conform to the assumptions specified in Section 3.1.1. In the discussion below, while we will occasionally refer to the results from the test data set DS-TEST-2, we will primarily focus on the results from the data set DS-TEST-1 as the results from the two data sets exhibit a fairly similar pattern for recognition accuracy.

Baseline and Ambiguity Detection. The recognition accuracy of the video text recognizer (VTR) for test data set `DS-TEST-1`, without the ambiguity detection and the audio-video (A/V) combination, was found to be 53.7%. This is shown in Table 21 as row 1 and serves as the baseline for future comparisons. The baseline VTR accuracy for the test data set `DS-TEST-2` was found to be 58.9% and this is shown in Table 21 as row 1. For the test data set `DS-TEST-1`, as a result of ambiguity detection 64.3% of the characters were classified as ambiguous, while the remaninig 35.7% were classified as non-ambigous. The accuracy for the set of ambiguous characters, S_D was found to be 39.2% while the accuracy for the set of non-ambiguous character, $S - S_D$ was found to be 79.8%. For the test data set `DS-TEST-2`, the set S_D constituted 59.8% of the entire data set and had an accuray of 44.6%. The corresponding values for the set $S - S_D$ were 40.2% and 80.1%.

Rank Based Techniques. Rows 2-4 in Table 21 and Table 22 list the results from the use of different rank sum based techniques for A/V combination. The rank sum for these experiments was based on different subsets of audio features F1, F2, F3 and the video text recognizer match score (denoted using V). For the test data set `DS-TEST-1`, we observe that the rank sum that was based on V and F1 has the highest end-to-end system accuracy of 60.7% for this technique. Notice that the other audio features do not seem to significantly impact the combination stage, which can be attributed to the fact that the test sets `DS-TEST-1` and `DS-TEST-2` are reasonably well aligned in terms of the handwritten and the spoken content.

Recognizer-Specific Weight Based Techniques. For recognizer-specific weights (Recognizer Wts.), we used different combinations of $[w_V, w_A]$ and the corresponding results are shown as rows 5-13 in Table 21 and Table 21. For the test data set `DS-TEST-1`, experimental results show that the optimal value of weights is close to $[0.8, 0.2]$, with an end-to-end accuracy $\alpha(S) = 65.1\%$. It is interesting to see how the four ratios corresponding to the ambiguous characters that are correctly/incorrectly

Table 21: Combination results for test data set `DS-TEST-1`

#	Technique	Non-Ambiguous Chars. $S - S_D$			Ambiguous Chars. S_D						All Chars. S
		$\frac{\#V_C}{\lvert S\rvert}$	$\frac{\#V_W}{\lvert S\rvert}$	$\alpha_V(S - S_D)$	$\frac{\#(V_C, AV_C)}{\lvert S\rvert}$	$\frac{\#(V_C, AV_W)}{\lvert S\rvert}$	$\frac{\#(V_W, AV_C)}{\lvert S\rvert}$	$\frac{\#(V_W, AV_W)}{\lvert S\rvert}$	$\alpha_V(S_D)$	$\alpha_Z(S_D)$	$\alpha(S)$
		%	%	%	%	%	%	%	%	%	%
1	VTR ★	53.7	46.3	53.7	–	–	–	–	–	–	53.7
	RankSum										
2	$[V, F1, 2, 3]$	28.5	7.2	79.8	19.0	6.2	8.5	30.6	39.2	42.8	56.0
3	$[V, F1, 2]$	”	”	”	20.5	4.8	8.7	30.4	”	45.3	57.6
4	$[V, F1]$■	”	”	”	19.6	5.6	12.6	26.4	”	50.2	60.7
	Rec. Wts.										
5	$[0.1, 0.9]$	28.5	7.2	79.8	18.4	6.8	14.3	24.8	39.2	50.8	61.2
6	$[0.2, 0.8]$	”	”	”	18.6	6.6	14.0	25.0	”	50.8	61.2
7	$[0.3, 0.7]$	”	”	”	18.6	6.6	14.0	25.0	”	50.8	61.2
8	$[0.4, 0.6]$	”	”	”	19.2	6.0	14.0	25.0	”	51.8	61.8
9	$[0.5, 0.5]$	”	”	”	19.6	5.6	14.0	25.0	”	52.4	62.2
10	$[0.6, 0.4]$	”	”	”	20.2	5.0	14.0	25.0	”	53.4	62.8
11	$[0.7, 0.3]$	”	”	”	21.5	3.7	13.4	25.6	”	54.3	63.4
12	$[0.8, 0.2]$♠	”	”	”	23.1	2.1	13.4	25.6	”	56.9	65.1
13	$[0.9, 0.1]$	”	”	”	24.8	0.4	9.9	29.1	”	54.0	63.2
	Chr. Wts.										
14	Sensitivity	28.5	7.2	79.8	19.6	5.6	14.0	25.0	39.2	52.4	62.2
15	Precision ♣	”	”	”	18.4	6.8	17.6	21.5	”	55.9	64.5
	Ensemble										
16	■, ♠, ♣	28.5	7.2	79.8	–	–	–	–	39.2	54.9	63.8
17	★, ■, ♠, ♣	”	”	”	–	–	–	–	”	58.2	65.9
18	★, ♠, ♣	”	”	”	–	–	–	–	”	59.0	66.4

V_C: Correct VTR Output, V_W: Wrong VTR Output, AV_C: Correct A/V Combination Output, AV_W: Wrong A/V Combination Output

Table 22: Combination results for test data set `DS-TEST-2`

#	Technique	**Non-Ambiguous Chars.** $S - S_D$			**Ambiguous Chars.** S_D						**All Chars.** S
		$\frac{\#V_C}{\lvert S\rvert}$	$\frac{\#V_W}{\lvert S\rvert}$	$\alpha_V(S - S_D)$	$\frac{\#(V_C, AV_C)}{\lvert S\rvert}$	$\frac{\#(V_C, AV_W)}{\lvert S\rvert}$	$\frac{\#(V_W, AV_C)}{\lvert S\rvert}$	$\frac{\#(V_W, AV_W)}{\lvert S\rvert}$	$\alpha_V(S_D)$	$\alpha_Z(S_D)$	$\alpha(S)$
		%	%	%	%	%	%	%	%	%	%
1	VTR ★	58.9	41.1	58.9	–	–	–	–	–	–	58.9
	RankSum										
2	$[V, F1, 2, 3]$	32.2	8.0	80.1	17.8	8.9	9.0	24.1	44.6	44.8	59.0
3	$[V, F1, 2]$	”	”	”	19.6	7.1	9.4	23.7	”	48.5	61.2
4	$[V, F1]$■	”	”	”	18.5	8.2	11.2	21.9	”	49.7	61.9
	Rec. Wts.										
5	$[0.1, 0.9]$	32.2	8.0	80.1	17.3	9.4	12.7	20.4	44.6	50.2	62.2
6	$[0.2, 0.8]$	”	”	”	17.3	9.4	12.7	20.4	”	50.2	62.2
7	$[0.3, 0.7]$	”	”	”	17.6	9.1	12.6	20.5	”	50.5	62.4
8	$[0.4, 0.6]$	”	”	”	17.6	9.1	12.6	20.5	”	50.5	62.4
9	$[0.5, 0.5]$	”	”	”	19.5	7.2	12.0	21.1	”	52.7	63.7
10	$[0.6, 0.4]$	”	”	”	20.4	6.3	12.2	20.9	”	54.5	64.8
11	$[0.7, 0.3]$	”	”	”	22.5	4.2	11.4	21.7	”	56.7	66.1
12	$[0.8, 0.2]$♠	”	”	”	24.6	2.1	10.5	22.6	”	58.7	67.3
13	$[0.9, 0.1]$	”	”	”	25.9	0.8	7.7	25.4	”	56.2	65.8
	Chr. Wts.										
14	Sensitivity	32.2	8.0	80.1	19.2	7.5	12.3	20.8	44.6	52.7	63.7
15	Precision ♣	”	”	”	18.6	8.1	17.1	16.0	”	59.7	67.9
	Ensemble										
16	■, ♠, ♣	32.2	8.0	80.1	–	–	–	–	44.6	55.0	65.1
17	★, ■, ♠, ♣	”	”	”	–	–	–	–	”	57.0	66.3
18	★, ♠, ♣	”	”	”	–	–	–	–	”	61.4	68.9

V_C: Correct VTR Output, V_W: Wrong VTR Output, AV_C: Correct A/V Combination Output, AV_W: Wrong A/V Combination Output

recognized by the video text recognizer and the combined recognizer, change when the weights are varied. For the set of ambiguous characters S_D, V_C is used to denote the characters that were correctly recognized by the video text recognizer and V_W is used to denote the characters incorrectly recognized by the video text recognizer. Similar notation, AV_C and AV_W, is used for the set of characters correctly and incorrectly recognized by the A/V combination component. For high values of w_V, the final A/V combination output tends to be very similar to the video text recognizer output and results in a high ratio for (V_C, AV_C) and (V_W, AV_W) and a much smaller ratio of characters change from V_C to AV_W or V_W to AV_C. As expected, the reverse trend is true when w_A takes a higher value. The improvement in the final result comes from reducing the number of characters in (V_C, AV_W).

Character-Specific Weight Based Techniques. In the experiments conducted for character-specific weights (Character Wts.), shown as rows 14-15 in Table 21 and Table 22, we have used two ways to determine these weights. The first uses the video text recognizer sensitivity of each character as w_V and the second used the video text recognizer precision of each character as w_V. Here, we have used $w_A = 1 - w_V$. For the two techniques evaluated here, the technique that uses precision as the video weight performs better than the technique that use sensitivity. For the test data set `DS-TEST-1` the $\alpha(S)$ was found to be 64.5%. Here, the improvement mostly comes from an increased number of characters in (V_W, AV_C). Similar behavior was exhibited by the results from the experiments that used the test data set `DS-TEST-2`.

Recognizer Ensemble Based Techniques. Finally for ensemble based techniques, results reported in rows 16-18 of Table 21 and Table 22 demonstrate the need to select the correct set of recognizers for the ensemble. The results also demonstrate that increasing the number of recognizers does not necessarily increase the final recognition accuracy $\alpha(S)$. For ensemble based techniques, we see a significant improvement in the end-to-end character recognition accuracy. For the test data set `DS-TEST-1`, the

maximum accuracy achieved was 66.4% which is 12.7% absolute improvement and a 23.6% relative improvement compared to the baseline video text recognizer accuracy.

6.5 Summary

In this chapter we presented and evaluated a number of audio-video combination techniques that could be used by the end-to-end recognition proposed in this dissertation. In the context of audio-video based handwritten mathematical content recognition from classroom videos, it is important to note that while, at first glance, the combination problem looks largely intractable and very challenging; the use of ambiguity detection and option selection component and the synchronization component makes it possible to incorporate some well-know classifier combination techniques into the end-to-end recognition system. This chapter described how such combination techniques can be customized to incorporate various nuances (*e.g.* character-specific weights, etc.) into the recognition system. Experiments presented in this chapter demonstrated the improvement in end-to-end character recognition accuracy that can achieved by using the proposed recognition system.

CHAPTER VII

GRAMMAR ASSISTED AUDIO-VIDEO BASED MATHEMATICAL CONTENT RECOGNITION

7.1 Introduction

There is an abundance of research [13, 28, 7, 126] that makes use of grammar for mathematical content recognition from printed and/or handwritten documents. The use of an appropriate grammar during the recognition process helps to improve not only the character recognition accuracy but also the structure recognition accuracy associated with the mathematical content. In general, recognition systems that utilize grammar have a better recognition accuracy compared to systems that are grammar agnostic [81]. However, the use of grammar often limits the space of recognizable content to the content that can be expressed using the grammar. In this chapter, we investigate the use of grammar for audio-video based handwritten mathematical content recognition. Towards this end, we propose a multi-step technique consisting of: (1) *Initial Recognition* - where a video text recognizer and a speech recognizer, configured with a *base mathematical speech grammar*, are used to determine character and speech recognition results using the handwritten and the spoken content, (2) *Constrained Speech Grammar Generation* - where techniques based on the initial recognition results and the base mathematical speech grammar are used to generate a *constrained speech grammar* that is specific to the content being recognized, and (3) *Final Recognition* - where using a speech recognizer, configured with the constrained speech grammar, the final character recognition results are determined and a subsequent layout analysis is used to generate the final recognized content.

In the past, researchers have proposed several approaches for mathematical content recognition, which, in general, consists of two primary tasks - character recognition and structure recognition. While a number of the proposed approaches assume the presence of character recognition results and focus solely on recognizing the structure associated with the mathematical content, some approaches are integrated closely with the character recognition techniques and attempt to exploit any feedback the structure recognition component may have on the character recognition results. Prominent among the structure recognition techniques are the rule based approaches [62], methods based on projection profiles [75, 87] and several grammar based approaches [13, 28]. In recent years, grammar based techniques have emerged as the preferred approach for the recognition of mathematical content. Akio *et al.* [28], for instance, use a formal grammar for the verification of recognized mathematical equation. Similarly, Celik *et al.* [13] make use a probabilistic context-free graph grammar to find mathematically valid interpretations and probabilities associated with each such interpretation of recognition results from a character recognizer. The grammar assisted audio-video based technique presented in this chapter, while similar in spirit to the grammar-based approaches described above, is the first to combine traditional character recognition with speech grammar and speech recognition for handwritten mathematical content recognition. Specifically, we make the following contributions:

- *Improved Character Recognition Accuracy*, which results from the fact that all the strings generated using the constrained speech grammar, by virtue of being derived from the initial recognition results and the base mathematical speech grammar, have the correct number of utterances and have a higher chance of containing the correct utterances.

- *Simplified Audio-Assisted Structure Recognition*, which utilizes the cues from the recognition results generated by the speech recognizer (*e.g.* disambiguating

between the digit '2' and the exponent 'square') to assist in structure recognition, and information about segmented characters to aid in construction of parse tree for layout analysis.

The remainder of this chapter is organized as follows. Section 7.2 describes the key assumptions with regard to the characteristics of the video. Section 7.3.1 presents an overview of the solution in the context of the end-to-end system and describes the speech grammar that defines the audio content, which is assumed to accompany the handwritten content in a classroom video. Details of the proposed grammar-based recognition technique are discussed in Section 7.4. Experimental evaluation is presented in Section 7.5 and a summary of key findings from this chapter is presented in Section 7.6.

7.2 Assumptions

The solution proposed in this chapter for mathematical content recognition makes certain assumptions about the characteristics of the input video and the preprocessing stage. These include (1) the entire handwritten mathematical content is spoken and the spoken content is assumed to conform to a known *base mathematical speech grammar*. In the data set used for evaluation, the audio content accompanying the handwritten content conforms to the base mathematical speech grammar specified in Section 7.3.2, (2) the video timestamp determined for the segmented characters is known correctly to the extent such that the correct order of their actual appearance on the whiteboard is maintained, and (3) it is further assumed that the video segment containing the handwritten mathematical content to be recognized is known in advance and the corresponding audio content, which is assumed to be in the video segment, contains only the utterances that correspond to the handwritten mathematical content.

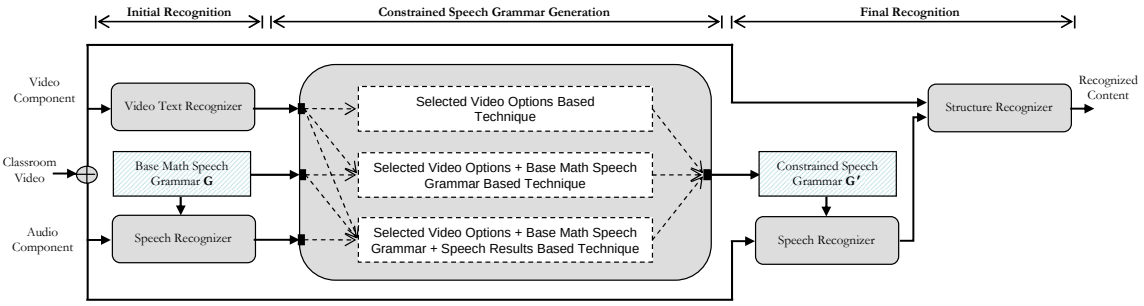


Figure 29: Overview of the grammar assisted audio-video based mathematical content recognition solution

7.3 Solution Overview & Base Mathematical Speech Grammar

In this section, we present an overview of the solution proposed in this chapter. This is followed by a description of the *base mathematical speech grammar*.

7.3.1 Solution Overview

The solution proposed in this chapter, shown in Figure 29, attempts to iteratively refine the recognition results through the various steps of the solution. The first step of the solution could be considered akin to a human glancing at the content being handwritten while also paying some attention to the content being spoken by the instructor. This step, termed initial recognition, produces character recognition results for each segmented character of the handwritten mathematical content using a video text recognizer and the corresponding speech recognition results determined using a speech recognizer (Sphinx-4 based audio text recognizer described in Section 2.4.2), configured with a *base mathematical speech grammar* **G**. Since these are still the initial recognition results, there may be one or more possible interpretations corresponding to a given segmented character. Note that the A/V combination component, described in Chapter 6, can be easily configured to supply multiple options for each segmented character and could therefore be used in place of the video text recognizer component in this solution.

The second step of the solution uses the base mathematical speech grammar $\mathbf{G}$ and the initial recognition results (both character and speech recognition results) to determine a *constrained speech grammar* $\mathbf{G}'$ for the next step of the solution. For the purpose of generating the constrained speech grammar, one of the three proposed techniques could be used. The first technique relies solely on the character recognition results from the video text recognizer and, for each segmented character, makes use of the audio utterances corresponding to a selected subset of video options to generate the constrained speech grammar $\mathbf{G}'$. Moreover, the grammar $\mathbf{G}'$ is generated such that the number of utterances contained in any string is fixed and is based on the number of segmented characters in the mathematical content being recognized. The second technique, in addition to using the character recognition results as described above, makes use of the base mathematical speech grammar to ensure that the strings generated using the grammar $\mathbf{G}'$ conform to the grammar $\mathbf{G}$. The third technique attempts to improve the grammar $\mathbf{G}'$ generated by the second technique by using the initial speech recognition results. The technique augments the grammar $\mathbf{G}'$, if necessary, to ensure that if the initial speech recognition results contain the correct utterance for a specific segmented character, the same is included in the grammar even when the initial character recognition results do not correspond to the correct utterance.

The final step of the solution configures the speech recognizer to use the constrained speech grammar $\mathbf{G}'$ and the recognition result from the speech recognizer is then combined with the results from X-Y cuts based method [35] to generate the final recognized content.

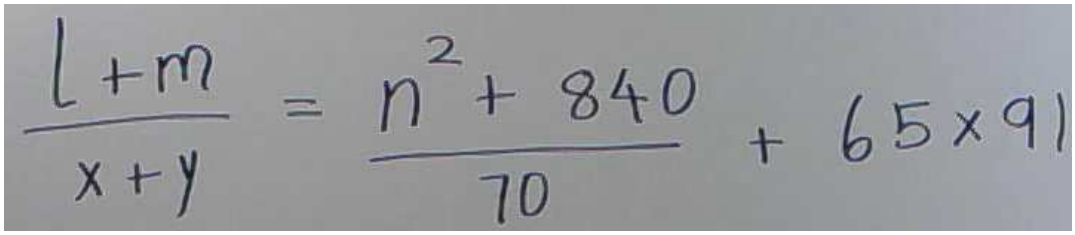
7.3.2 Base Mathematical Speech Grammar

A *Linguist*, as described in Section 2.4.2, is one of the three primary modules of the Sphinx-4 [98, 99] framework and is the module that is responsible for generation

Table 23: Base mathematical speech grammar

```
<equation>   = ( <expression> equals <expression> );
<expression> = ( <element> ( <symbol> <element> )* );
<symbol>     = ( plus | minus | divide by | multiplied by | into |
                 subtract | by );
<element>    = ( <algebra>  [ ( square | cube ) ] |
                 <number>   [ ( square | cube ) ] );
<algebra>    = ( a | b | c | d | e | f | g | h | i | j | k | l | m |
                 n | o | p | q | r | s | t | u | v | w | x | y | z );
<number>     = ( <digits> [ point <digits> ] );
<digits>     = ( zero | one | two | three | four | five | six |
                 seven | eight | nine )+;
```

of the *SearchGraph*. In generating the *SearchGraph*, the *Linguist* makes use of the language structure as represented by a given *LanguageModel* and the *AcousticModel*, which is the topological structure representing basic units of sound. In some cases, the *Linguist* may also use a *Dictionary*, which from the perspective of the Sphinx-4 framework is the translation of the words contained in the *LanguageModel* to appropriate sequences of *AcousticModel* elements. Sphinx-4 supports a number of formats for the specification of the *LanguageModel*. In the solution presented in this chapter, we make use of the Java Speech API Grammar Format (JSGF) [51], which defines a BNF-style, platform-independent, and vendor-independent unicode representation of grammars. We use JSGF to specify the *base mathematical speech grammar*, which is shown in Table 23. The spoken content from our classroom video dataset, which accompanies a handwritten content, conforms to this base mathematical speech grammar. We will use **G** to refer to the base mathematical speech grammar. Also note that although the base mathematical speech grammar is fairly general, it is still a regular grammar [1] and this is a feature that we exploit in one of the recognition techniques proposed later in this chapter.



(a) Handwritten equation from the whiteboard

Time (sec)	Utterance
5.5	l
6.0	plus
7.0	m
9.0	by
10.5	x
11.0	plus
12.0	y
14.0	equals

Time (sec)	Utterance
15.5	n
16.5	square
19.0	plus
19.5	eight
20.0	four
20.5	zero
22.5	divide by
24.5	seven

Time (sec)	Utterance
25.0	zero
26.0	plus
31.5	six
32.0	five
33.5	multiplied by
36.0	nine
36.5	one

(b) Manually labeled and timestamped audio utterances corresponding to the handwritten equation above

Figure 30: Section of the whiteboard showing handwritten content and the corresponding timestamped audio utterances from the classroom video

7.4 Recognition Technique

As outlined in the previous section, our technique for grammar assisted audio-video based handwritten mathematical content recognition consists of three steps. We will make use of a running example to assist in the explanation of the proposed technique. Figure 30 shows a section of the whiteboard containing the handwritten content and it also shows the timestamped sequence of audio utterances in the classroom video that correspond to the handwritten content.

7.4.1 Initial Recognition

In this step, we make use of the video text recognizer and the Sphinx based speech recognizer, configured with the base mathematical speech grammar **G**, to arrive at initial character and speech recognition results that are further refined by the subsequent stages.

Timestamp (seconds)	Segmented Image	Recognition Result
6.3		(L,1.00) (I,0.76) (r,0.74)
7.7		(f,0.86) (+,0.85) (t,0.84)
9.4		(m,0.99) (n,0.84) (M,0.76)
10.6		(/,1.00)
12.2		(Y,0.90) (X,0.88) (f,0.79)
13.0		(f,0.95) (t,0.92) (r,0.83)
14.2		(Y,1.00) (1,0.98) (f,0.88)
16.3		(=,1.00)
18.4		(M,0.87) (n,0.86) (i,0.75)
18.8		(2,0.97) (7,0.90) (Z,0.88)
19.8		(f,0.86) (+,0.85) (i,0.84)
21.3		(8,1.00) (g,0.90) (R,0.71)

Timestamp (seconds)	Segmented Image	Recognition Result
22.4		(4,0.96) (i,0.84) (+,0.79)
23.6		(0,0.99) (O,0.99) (D,0.89)
25.1		(/,1.00)
27.4		(7,0.90) (1,0.88) (I,0.85)
28.1		(0,0.99) (O,0.99) (D,0.90)
29.6		(+,0.91) (t,0.85) (r,0.84)
33.2		(6,1.00) (b.0.81) (S,0.81)
35.1		(5,1.00) (S,0.89) (f,0.82)
36.3		(X,0.98) (f,0.84) (2,0.82)
39.1		(9,1.00) (q,0.98) (4,0.90)
39.2		(7,0.89) (i,0.84) (1,0.83)

Figure 31: Recognition results from the video text recognizer

The video text recognizer operates on each segmented character $s \in S$ from the handwritten content under consideration to generate the video options. Continuing with the notation used in previous chapters, for each segmented character s the number of generated video options, *i.e.* $|V(s)|$ is equal to the cardinality of the dictionary C, and each video option $\mathbf{v}_j(s)$ consists of the character name $v_j^c(s)$ and the video match score $v_j^p(s)$. For the handwritten content shown in Figure 30, a selected subset of video options, for each segmented character, is shown in Table 31. Note that instead of using the results from the video text recognizer, an ordered list of character recognition results corresponding to each segmented character can be obtained from the A/V combination component as well.

As compared to the word-spotter that was used in the previous chapters, the output of the Sphinx-4 based audio text recognizer is independent of the video options from the video text recognizer. The speech recognizer makes use of the base mathematical speech grammar $\mathbf{G}$ to recognize the audio content corresponding to

Time	Utterance
6.6	f
7.2	into
7.5	m
10.1	by
10.3	eight
10.8	by
12.0	x
12.3	plus
12.5	y
13.3	by

Time	Utterance
14.5	v
14.7	plus
17.1	n
17.3	square
20.4	plus
20.7	eight
20.9	four
21.2	zero
23.7	divide by
25.5	seven

Time	Utterance
26.0	zero
27.3	plus
32.8	six
33.2	by
34.1	nine
34.9	divide by
37.0	two
37.4	nine
37.7	one

Figure 32: Recognition results from the Sphinx-4 based audio text recognizer

the handwritten content under consideration. The output is an ordered set of triplets represented as $\mathbf{x}_j$, where j represents the index at which the triplet is located in the ordered set. The triplet, which is a feature vector,

$$\mathbf{x}_j = [x_j^c, x_j^{t_{begin}}, x_j^{t_{end}}] \tag{52}$$

consists of x_j^c, which is the audio utterance corresponding to a character from the dictionary C (audio utterances appear as terminals in the grammar $\mathbf{G}$), and $x_j^{t_{begin}}$ and $x_j^{t_{end}}$, which denote the start and end times that delimit the occurrence of the utterance x_j^c in the audio content. We use $x_j^t = (x_j^{t_{begin}} + x_j^{t_{begin}})/2$ as an approximation of the time at which the terminal is assumed to have been found in the audio content. For our running example, Figure 32 shows the output of the Sphinx-4 based audio text recognizer when configured with the base mathematical speech grammar.

7.4.2 Generating the Constrained Speech Grammar

In this step, we make use of the recognition results from the previous step and the base grammar $\mathbf{G}$ to determine a constrained speech grammar $\mathbf{G}'$ such that it is more specific to the mathematical content being recognized. The following sections describe

three specific methods for generation of the constrained speech grammar.

7.4.2.1 Using selected video options

This method for the generation of the constrained speech grammar $\mathbf{G}'$ relies solely on the character recognition results returned by the video text recognizer. As a consequence, the strings that could be generated using the grammar $\mathbf{G}'$ do not necessarily conform to the base mathematical speech grammar $\mathbf{G}$. In this method, for the set of video options generated for each segmented character, we use selection criteria, similar to the ones used for ambiguity detection and option selection techniques, to determine a subset of video options for each segmented character. The audio utterances corresponding to the selected video options are then composed into a grammar such that (1) any string generated using the grammar has a fixed number of utterances, one for each segmented character, and these utterances appear in the same order as the segmented characters (arranged in increasing order of their timestamps), and (2) in any string generated using the grammar, the choice of utterance for any segmented character does not impact the choice of utterance for any other segmented character. For instance, when using top-3 video options (*i.e.* `NumOpt`=3, in terms of option selection methods proposed in Section 4.3) for each segmented character from our running example: (1) the number of utterances that should be contained in any string generated using the grammar $\mathbf{G}'$ is 23, which is the number of segmented characters present in our running example, and (2) in any string generated by the grammar $\mathbf{G}'$ there should be an utterance corresponding to each segmented character and for a given segmented character, any utterance corresponding to the top-3 video options can appear in the string and is independent of the utterances chosen for other segmented characters. The resulting grammar is shown in Table 24.

Although the speech recognition using the grammar $\mathbf{G}'$ happens as part of the final recognition step, we discuss the recognition results for ease of relating the results to

Table 24: Constrained speech grammar generated using the top-3 video options

```
<equation> = (l|i|r)(f|plus|t)(m|n)(divide by|by)(y|x|into|multiplied by|f)
             (f|t|r)(y|one|f)(equals)(m|n|i)(two|square|seven|z|zee)
             (f|plus|i)(eight|g|r)(four|i|plus)(zero|o|d)(divide by|by)
             (seven|one|i)(zero|o|d)(plus|t|r)(six|b|s)(five|s|f)
             (x|into|multiplied by|f|two|square)(nine|q|four)(seven|i|one);
```

the generated grammar $\mathbf{G}'$. The results from the video text recognizer and the speech recognizer, configured with grammar $\mathbf{G}$, act as the baseline for demonstrating the improvement. For the handwritten mathematical content shown in Figure 30, the top options generated by the video text recognizer correspond to the string `'L f m / Y f Y = M 2 f 8 4 0 / 7 0 + 6 5 X 9 7'`, while the result generated by the Sphinx-4 based audio text recognizer with just the base mathematical speech grammar correspond to the string `'f into m by eight by x plus y by v plus n square plus eight four zero divide by seven zero plus six by nine divide by two nine one'`. In comparison, the Sphinx-4 based audio text recognizer when configured with the grammar $\mathbf{G}'$ specified in Table 24 produces the following output: `'i f n by x f y equals n square plus eight four zero divide by seven zero plus six five multiplied by nine one'`. Even with this simple constrained grammar generation approach, the advantage of using a combination of video and audio based recognizers is clearly evident.

The selection of video options that are used for generating the grammar $\mathbf{G}'$ plays an important role in determining the accuracy of the final recognized content. Fortunately, this selection procedure is along the same lines as the ambiguity detection and option selection techniques as described in Chapter 4. Experiments, presented later, make use of top-n and relative thresholding techniques for video option selection.

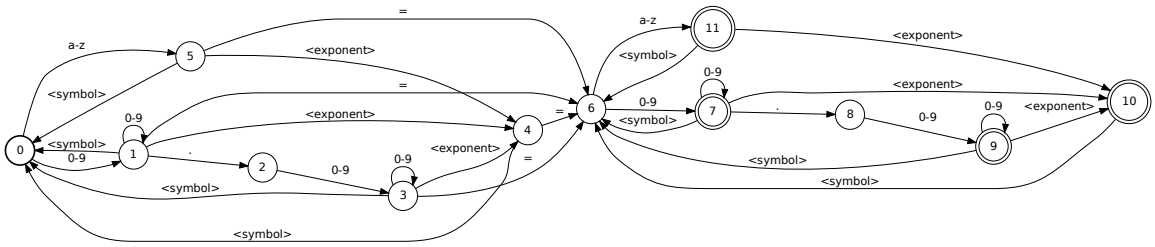


Figure 33: Finite state automaton corresponding to the base mathematical speech grammar

7.4.2.2 Using selected video options with base mathematical grammar

This method is similar to the one proposed above, with the added constraint that any string generated using the constrained speech grammar $\mathbf{G}'$ should conform to the base mathematical speech grammar $\mathbf{G}$. The added constraint, as will be shown in Section 7.5, results in increased recognition accuracy.

To generate the grammar $\mathbf{G}'$, in addition to using the selected video options, we make use of an automaton that represents the base mathematical speech grammar $\mathbf{G}$. Given the fact that the grammar $\mathbf{G}$ is a regular grammar we can make use of a finite state automaton to represent the grammar. The corresponding automaton for grammar $\mathbf{G}$, which is shown in Figure 33, consists of one start state (0) and four accepting states (7, 9, 10 and 11). The transition between the various states and the tokens that result in such transition are shown by arcs and the corresponding labels. In our case, the finite state automaton is capable of accepting any mathematical content that conforms to the grammar $\mathbf{G}$, *i.e.* beginning with state 0 as the initial state, the transitions that occur as a result of characters read sequentially from a given mathematical formulation would end at one of the accepting states. Any sequence of characters that does not conform to the grammar $\mathbf{G}$ either requires a non-existent transition or ends in a state that is not an accepting state. The procedure for generating the grammar $\mathbf{G}'$ using the finite state automaton and the selected video options is described next.

The three phase procedure for generating the grammar $\mathbf{G}'$ starts by selecting the

video options (*e.g.* top-n) for each segmented character. The automaton is initialized to the start state and the audio equivalent of the video options for the first character from the handwritten content are independently tried as input to the automaton. The audio equivalents that do not have a corresponding transition from the start state in the automaton for the base mathematical speech grammar are eliminated. The set of states that result from the first transition are referred to as the *level-1* states. The states along with the specific transitions (*i.e.* the specific audio equivalents) leading to them are recorded. The *level-1* states act as the starting states for the audio equivalent of the selected video options for the second character. All audio equivalents corresponding to the selected video options of the second character are tried for all the *level-1* states and the set of states that result from allowed transitions are referred to as the *level-2* states and act as the starting state for the next character. An important point to note here is that at any level the number states will not be more than the states contained in the finite state automaton corresponding to the grammar **G**. This is a direct result of fact that the base mathematical speech grammar **G**, described in Section 7.3.2, is a regular grammar and this limits an otherwise exponential increase in the number of states at any level. The process determining allowed transition, as described above, is repeated until all the characters from the handwritten mathematical content have been exhausted. At the end of this step we have the start state and the set of states and transitions from *level-1* to *level-m*, where m is the number of characters in the handwritten mathematical content. Next, the start state is assigned an identifier 0 and the remaining states are assigned consecutive identifiers starting with 1. The states and transitions are then combined to form a new automaton where 0 is the start state and set of states from *level-m* are the only ones designated as the accepting states. In the second phase of the procedure any states that do not have any possible path leading to the accepting state are eliminated. For the handwritten mathematical content shown in Figure 30, the

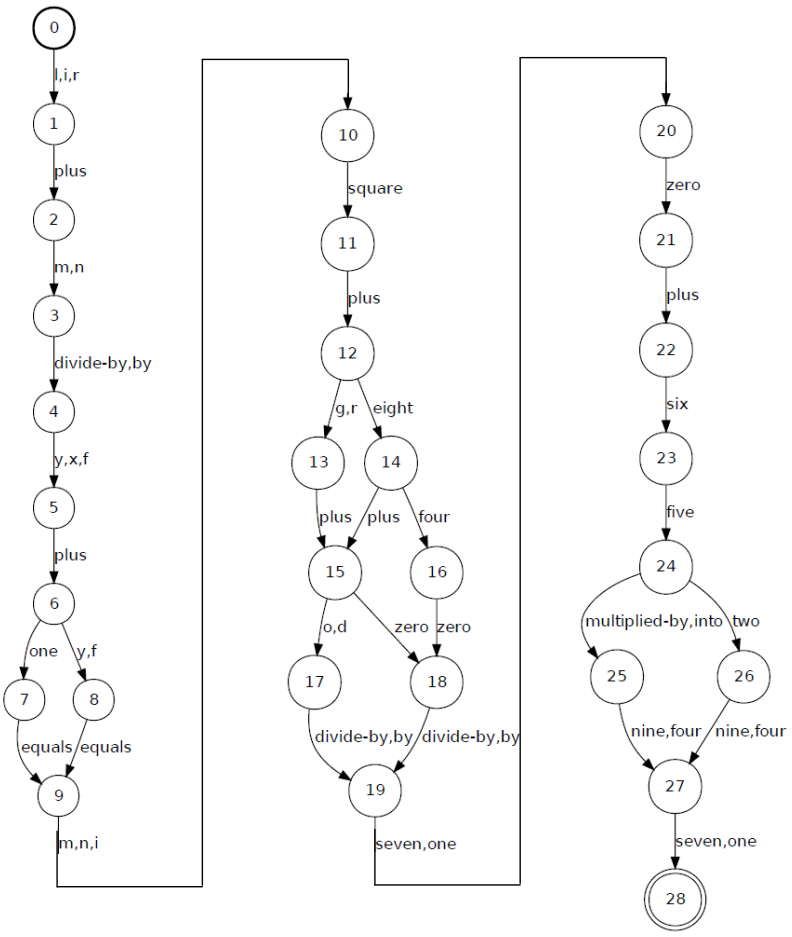


Figure 34: Finite state automaton corresponding to the constrained speech grammar

automaton constructed using the top-3 video options, the base mathematical speech grammar and the procedure described above is shown in Figure 34. In the final phase of the procedure, the automaton is translated to the regular grammar equivalent and this grammar acts as the constrained grammar $\mathbf{G}'$. The resulting constrained grammar is shown in Table 25. Note that in some cases, for a given set of states and a given character from the handwritten content, there may be no transition possible using the audio equivalents of the selected video options. In such cases, the procedure continues by including more video options until at least one valid transition is found.

Again, although speech recognition using the generated grammar $\mathbf{G}'$ is not a part of this step of the solution, we present the recognition results for ease of relating the

Table 25: Constrained speech grammar generated using the top-3 video options and the base mathematical speech grammar

```
<equation> = (l|i|r)(plus)(m|n)(divide by|by)(y|x|f)(plus)(y|f|one)(equals)
             (m|n|i)(square)(plus)(((g|r)(plus)(zero|o|d))|((eight)((plus)
             (zero|o|d)|(four)(zero))))(divide by|by)(seven|one)(zero)(plus)
             (six)(five)(multiplied by|into|two)(nine|four)(seven|one);
```

results to the grammar $\mathbf{G}'$. The Sphinx-4 based audio text recognizer when configured with the grammar specified in Table 24 produces the following output: `'i plus n by x plus y equals n square plus eight four zero divide by seven zero plus six five multiplied by nine one'` for the audio corresponding to the handwritten content shown in Figure 30. One can easily notice the improvement due to the use of grammar when compared to the results from the selected options based technique proposed in the previous section, which recognized the handwritten mathematical content as: `'i f n by x f y equals n square plus eight four zero divide by seven zero plus six five multiplied by nine one'`.

7.4.2.3 Using audio-video options with base mathematical speech grammar - `mxn` *method*

So far, the methods proposed for determining the constrained speech grammar $\mathbf{G}'$ have relied on the results from the video text recognizer and on the base mathematical speech grammar $\mathbf{G}$. The audio-video options based method proposed in this section, besides incorporating the results from the video text recognizer, also attempts to incorporate the recognition results from the speech recognizer to arrive at the constrained speech grammar $\mathbf{G}'$. The motivation behind including the results from the Sphinx-4 based audio text recognizer in the determination of the constrained speech grammar is to: (1) improve the recognition accuracy by augmenting, if needed, the grammar with utterances from the initial speech recognition results, and (2) reduce the number of strings that could be generated using the grammar $\mathbf{G}'$ by pre-recognizing segmented characters that are recognized as the same character by the

video text recognizer and the Sphinx-4 based audio text recognizer. The reduction in the number of strings becomes important when the base mathematical speech grammar is more complex and ensuring that the constrained speech grammar conforms to the base mathematical speech grammar involves generation and validation of all strings that are possible using the selected video options.

The task of incorporating the results from the video text recognizer and the Sphinx-4 based audio text recognizer into the grammar $\mathbf{G}'$ involves combining the output of the two recognizers and is faced with audio-video synchronization issues. While some of the audio-video synchronization issues are similar to the ones described in Chapter 5, there are specific challenges that can be attributed to the use of speech recognizer. For instance, besides incorrect recognition, a single segmented character could be recognized as multiple segmented characters by the speech recognizer (*e.g.* in some cases utterance `'equals'` is recognized as `'eight plus'` by the speech recognizer, which corresponds to two segmented characters) or a segmented character may not appear in the speech recognition result.

Our approach for audio-video synchronization and the generation of constrained speech grammar, called the `mxn` method, relies on using a sequence of `m` words from the output of the speech recognizer and locating the occurrence of this *audio word sequence* in the top-`n` character recognition results for a selected subset of segmented characters. Given a sequence of `m` words starting at the ith utterance from the result of the Sphinx-4 based audio text recognizer, we use $t^a_{begin} = x_i^{t_{begin}}$ to denote the start time corresponding to the utterance of the first word in the sequence and use $t^a_{end} = x_{i+m}^{t_{end}}$ to denote the end time corresponding to the utterance of the last word in the sequence. Now, the top-`n` video options corresponding to any segmented character whose video timestamp is within the interval $[t^a_{begin} - \delta t, t^a_{end} + \delta t]$ are considered a potential candidate for the purpose of synchronization with the audio segment. If for a given audio word sequence of `m` words the number of selected segmented

characters is less than `m`, then we choose segmented characters from the vicinity of the already selected segmented characters to have at least `m` segmented characters. The process of synchronization (*i.e.* locating the sequence of `m` utterances from the speech recognition results amongst the top-`n` selected video options) is described next.

From among the set of segmented characters selected above, the first `m` characters are chosen and all possible combination of video options, while maintaining the temporal ordering, are generated. The audio utterance corresponding to each such combination of video options is generated and this is matched against the audio word sequence. If a match is found, then, for the corresponding set of segmented characters the set of video options is trimmed to one video option that corresponds to the audio utterance that matched the audio word sequence. If a match is not found, then the next set of `m` segmented characters is chosen starting at the second character and the process continues, choosing `m` segmented characters starting at the third character and so on, until a match is found or there are no more segmented characters left in the selected subset. As for the audio word sequence, if a match is found then the next set of `m` words is chosen starting where the last sequence of `m` words ended, else the next set of `m` words is chosen leaving the first word from the last audio word sequence.

In our implementation of the `mxn` method, we have implemented an optimization that keeps track of the segmented characters that have already been successfully synchronized to the results from the speech recognizer. Such segmented characters are not considered for synchronization again. For segmented characters that are not synchronized but they lie between two synchronized segmented characters and there is a 1-1 correspondence between the non-synchronized segmented characters and the non-synchronized utterances, the set of `n` video options for such non-synchronized segmented characters is augmented to contain the video equivalent of the audio utterance as a video option (if the video option isn't already included).

The video options resulting from the process defined above, in general, have lesser

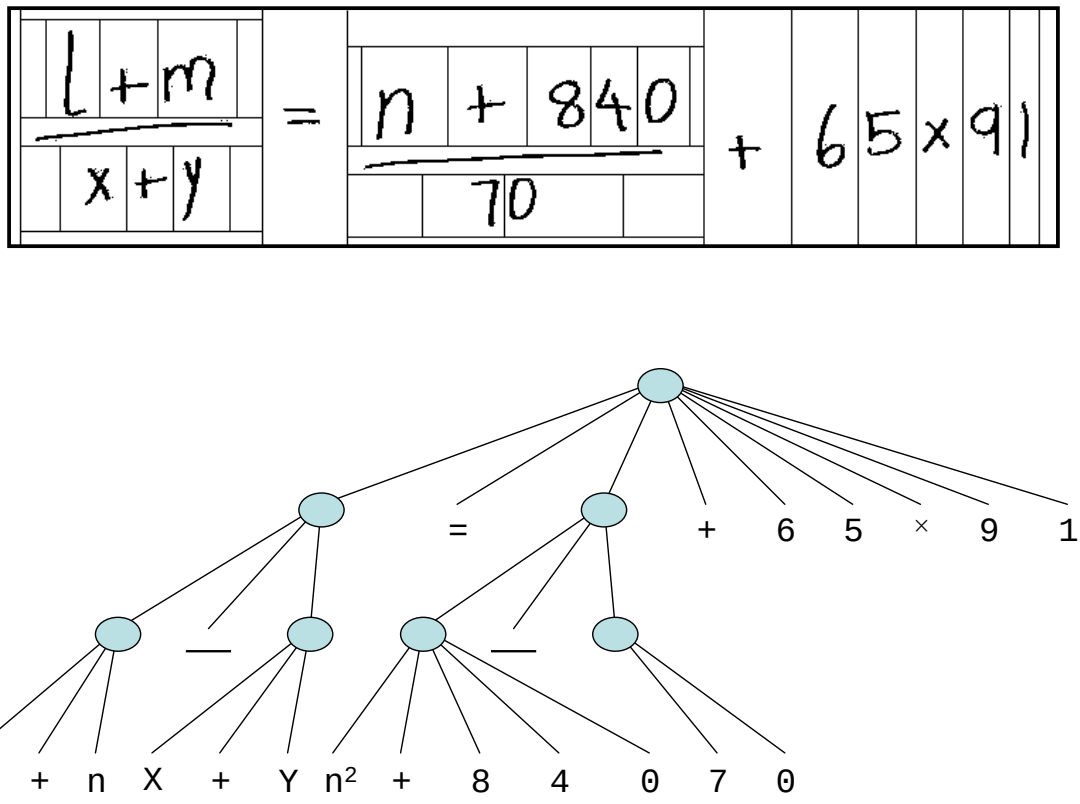


Figure 35: Image of the handwritten mathematical content after X-Y cuts and the corresponding parse tree

number of video options per segmented character. These video options and the base speech grammar are then used in a manner similar the one described in Section 7.4.2.2 to arrive at the constrained speech $\mathbf{G}'$. Note that in cases where the base mathematical speech grammar $\mathbf{G}$ is not a regular grammar, one may need to generate a grammar similar to what was generated in Section 7.4.2.1, generate all possible strings using the grammar, check their conformance to the base mathematical speech grammar and generate the constrained speech grammar using the conforming strings. In such cases, having lesser number of video options per segmented character helps reduce the time taken (which may run into hours [13]) to arrive at the constrained speech grammar $\mathbf{G}'$.

7.4.3 Final Recognition

The final step of the recognition process, involves (1) configuring the Sphinx-4 based audio text recognizer with the constrained speech grammar $\mathbf{G}'$ and using it for recognizing the spoken content, and (2) using the recognition results from the Sphinx-4 based audio text recognizer in combination with layout results from a modified implementation of X-Y cuts [35] based method for recognizing the structure of the mathematical content.

Since the constrained speech grammar $\mathbf{G}'$ already incorporates the recognition results from the video text recognizer and, for some constrained grammar generation methods, also conforms to the base mathematical speech grammar, the process of recognizing the spoken content is fairly straightforward and the results are fairly accurate. For instance, experiments reported later in this chapter, show that our system is able to achieve a character recognition accuracy of 78.1% when using the constrained grammar generation method presented in Section 7.4.2.2.

While a number of well known structure recognition techniques [2, 126, 17, 3], which utilize the character recognition results could have been employed for the purpose of structure recognition at this stage of the solution, we wanted to experiment with the use of audio content in the structure recognition process, and examine how the speech recognition results can simplify some tasks associated with structure recognition and possibly improve the structure recognition accuracy as well. Towards this end, we examine the possibility of using the audio content for structure recognition in the context of the X-Y cuts based method [35] and make two important modifications to the method: (1) We make use of cues (e.g. `'square'`, `'cube'`, `'multiplied by'`, etc.) from the speech recognition results to assist in structure recognition and to simplify the X-Y cuts based method, and (2) The character segmentation results and a heuristic for handling skew associated with handwritten content is used to assist in the construction of the parse tree. The modified X-Y cuts based method is described

next.

To simplify the X-Y cuts and to demonstrate the use of audio in the structure recognition process, from the image of the handwritten mathematical content to be recognized, we remove the segmented character images corresponding to exponents (identified from the speech recognition results as `'square'` or `'cube'`) and modify to character recognition result for the preceding segmented character to contain the exponent. For instance, in our running example, we remove the exponent `'square'` that follows the character `'n'` and incorporate the exponent in the character recognition result for `'n'`.

To address issues related to skew associated with handwritten mathematical content, the proposed X-Y cuts based method exploits the fact that the recognition system is aware of the number of segmented characters and their location coordinates. Besides recording the image segment at each node of the X-Y parse tree, each node also records the number of segmented characters contained in the corresponding image segment. The X-Y cut algorithm is applied recursively until there is only one segmented character left within the image segment corresponding to every leaf node of the parse tree. This is shown in Figure 35. In some cases, however, due to skew and character mis-alignment associated with handwritten content the X-Y cut algorithm may not be able to proceed even when there is more than one segmented character remaining in the associated parse tree node. For such parse tree nodes we make use of a heuristic that gradually increases the spacing between the segmented characters and enables the cuts. This is achieved by scaling the image segment contained at the node by a factor $1 + \delta y$ along the Y dimension, when attempting Y cuts and scaling by a factor of $1 + \delta x$ along the X dimension for the X cuts. However, in doing so the size of the segmented characters is kept constant and only the *bottom-left* corner of a segmented character contained in the image segment is relocated to the new coordinates after scaling. For the parse tree node in question, the heuristic of scaling

along the X and Y dimension is alternated until an X or Y cut is possible. If the type of cut (X or Y) is the same as the one required at the parse tree node, then nodes resulting from the cut are added as children of the parse tree node in question. If the type of cut is not the one required at the parse tree node then the nodes resulting from cut are added as siblings of the node in question.

Once the parse tree construction is complete, the segmented characters at the leaf nodes (one at each leaf node) are replaced by the corresponding recognition results. The parse tree is then traversed to arrive at the final recognition result in LaTeX format. The final recognition result for our running example, using the top-n video options with base mathematical grammar approach, is shown below in LaTeX format,

```
\frac{I + n}{X + Y} = \frac{n^2 + 840}{70} + 65 \times 91
```

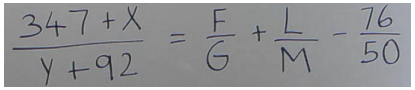
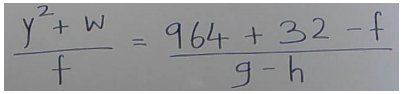
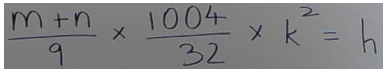
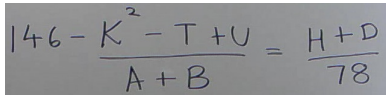
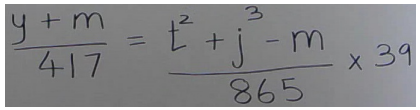
and the corresponding rendering of the LaTeX result is shown next.

$$\frac{I + n}{X + Y} = \frac{n^2 + 840}{70} + 65 \times 91$$

7.5 Experiments

This section presents the results from our evaluation of the mathematical content recognition technique proposed in this chapter. The evaluation was conducted over a set of 100 mathematical equations retrieved from the data sets `DS-TRAIN-1` and `DS-TEST-1`. It was ensured that the chosen equations were in conformance to the assumptions listed in Section 7.2. We start by presenting recognition results for a few sample equations, which is followed by results from the evaluation of the proposed constrained grammar generation methods using the complete data set of 100 equations.

Table 26: Extracted mathematical equations and the corresponding recognition results

S.No.	Image	Video Recognition Result*	Grammar Assisted A/V Recognition Result
1.		$\frac{J41+X}{YVq7} = \frac{F}{S} + \frac{L}{M} - \frac{76}{50}$	$\frac{J+7+X}{Y+91} = \frac{F}{S} + \frac{L}{M} - \frac{76}{50}$
2.		$\frac{Y^T+W}{+} = \frac{9b4+77-+}{9-h}$	$\frac{Y^2+W}{f} = \frac{964+32-f}{g-h}$
3.		$\frac{mfn}{9} \times \frac{1004}{7L} \times k^2 = h$	$\frac{m+n}{9} \times \frac{1004}{97} \times h^2 = h$
4.		$I46 - \frac{Y^t-T+V}{AfB} = \frac{H4D}{18}$	$I+6-\frac{k^2-T+U}{h+B} = \frac{H+D}{78}$
5.		$\frac{Y+m}{4)1} = \frac{G^7+J^7-M}{8b+} \times 39$	$\frac{Y+m}{411} = \frac{f^2+J^3-M}{867} \times 39$

*For ease of comparing the character recognition results, the recognition results are presented using the structure recognition results from the grammar assisted A/V combination technique.

7.5.1 Recognition results for sample equations

In this section we discuss the recognition results corresponding to 5 selected equations from our data set. The recognition results are based on the *selected video options with base mathematical grammar* method proposed in Section 7.4.2.2 and are intended to provide an insight into the corrections and errors introduced by the proposed technique. The video option selection used for the experiments was based on selecting top-n video options with `NumOpt`=3.

As is evident from the results presented in Table 26, the grammar assisted audio-video combination based technique is able to correctly recognize significantly more characters as compared to the video text recognizer. The corrections introduced by the proposed technique can be attributed to the use of a constrained speech grammar. The improvements due to the use of a grammar, for instance, are visible in equations corresponding to rows 3 and 4, where the recognition results from the video text recognizer `mfn` and `H4D` were subsequently correctly recognized as $\mathtt{m} + \mathtt{n}$ and $\mathtt{H} + \mathtt{D}$. Also, notice that in the equation corresponding to row 4, the recognition result from the video text recognizer `I46` was subsequently recognized as $\mathtt{I} + \mathtt{4}$, which is incorrect. In this case, the correct output was `146` and this is an example where grammar assisted audio-video combination approach can lead to a decrease in the character recognition accuracy. In the equation corresponding to row 2, the grammar assisted audio-video combination technique was able to correctly recognize `32`, which was incorrectly recognized as `77` by the video text recognizer. This correction can be attributed to the use of Sphinx-4 based audio text recognizer configured with the constrained speech grammar.

In the examples shown in Table 26, the structure recognition based on audio cues (for exponents in this case) and X-Y cuts is correct for all the equations. This can be attributed to the fact that the equations included in the data set are relatively less complex in terms of structure and do not include more complex mathematical

Table 27: Recognition accuracies for constrained grammar generation methods

Method	Char. Recog. Accuracy %	Struct. Recog. Accuracy
Video Text Recognizer (Baseline)	54.7	-
`NumOpt`=3	66.5	67/100
`OptRelThr`=0.90	68.2	68/100
`NumOpt`=3 + **G**	76.2	84/100
`OptRelThr`=0.90 + **G**	78.1	85/100
`mxn` method `m`=`n`=3 + **G**	75.8	84/100

notations like integrals, summations, etc. The examples, however, demonstrate that audio cues can be used to simplify the task of structure recognition. Such audio cues can be further extended to pre-recognize more complex structures like integrals, summations, etc. and help in the structure recognition process.

7.5.2 Evaluation of the constrained grammar generation methods

Table 27 presents the results from end-to-end evaluation conducted using the complete data set of 100 equations. The experiment was conducted once for each constrained grammar generation method (Section 7.4.2) proposed in this chapter. The results are analyzed in terms of character recognition and structure recognition accuracy (*i.e.* the LaTeX code is correct except for mistakes related to character recognition). We make use of the notation used in Section 4.3 for the option selection method. `NumOpt` denotes the case when top-n video options are used for constrained speech grammar generation, and `OptRelThr` denotes the case when relative threshold based option selection is used.

From the results, it is clear that the grammar assisted audio-video combination based technique results in a significant increase in the character recognition accuracy compared to the accuracy of the video text recognizer. While the improvement in the character recognition accuracy is highest when using the relative threshold based video option selection with base mathematical grammar, the `mxn` method also performs almost as well in improving the character recognition accuracy. One should note that

one significant advantage of the `mxn` method is the ability to limit the number of options for each segmented character, which subsequently helps in the generation of constrained speech grammar when the base grammar is not regular. For the structure recognition accuracy, again as expected the methods that take into account the base grammar outperform the top-3 video options based method. The lower structure recognition accuracy of the top-3 video options based method is tied to the character recognition accuracy. For instance, an incorrectly recognized exponent would lead to an incorrect structure recognition when using our approach.

7.6 Summary

In this chapter we presented a grammar assisted audio-video combination based technique for recognizing handwritten mathematical content from classroom videos. The approach attempts to validate the recognition results generated by a video text recognizer using a Sphinx-4 based audio text recognizer configured with a constrained speech grammar. We proposed different methods for generation of the constrained speech grammar, two of which ensure conformity to a chosen base mathematical speech grammar. We also proposed a way for incorporating audio cues into the layout analysis procedure, which can lead to improvement in the structure recognition accuracy and also lead to simplification of the layout analysis procedure. Experiments, conducted over a data set of 100 equations, were used to demonstrate that the use of proposed technique and the proposed constrained grammar generation methods can lead to significant improvements in the character and structure recognition accuracy associated with handwritten mathematical content from classroom videos.

CHAPTER VIII

RELATED WORK

The research presented in this dissertation lies at the intersection of three distinct and well-studied specializations associated with the field of digital signal processing. These specializations include text recognition, speech recognition and classifier combination. In the following sections we briefly describe the related work from each of these specializations.

8.1 Text Recognition

In this section, we start by providing a brief overview of the research in the field of handwriting recognition, and then we delve deeper into the topic of text extraction and recognition from videos followed by mathematical content recognition techniques.

8.1.1 Handwriting Recognition

There exists a vast collection of literature related to the recognition of handwritten text, Plamondon *et al.* [81] provide a very comprehensive survey of the research in this field. Handwriting recognition could be defined as the ability of a computer to interpret handwritten textual content from input sources which, amongst others, may include paper documents [23, 61, 88], pen-based inputs that write on an electronic screen [80], videos [104] and other specialized input devices [100]. Techniques for handwriting recognition can be categorized into two main categories, namely offline and online. The recognition is said to be *offline* when the input contains purely spatial pixel intensity information, as is the case with paper documents. In *online* recognition, it is assumed that we have spatio-temporal pixel intensity information

(*i.e.* stroke coordinates against time) and such information is used to facilitate recognition. While a number of methods for handwriting recognition start by recognizing individual characters [102], there is an equally large number of recognition systems that treat words as the smallest recognition entity [119]. The choice of a particular recognition method, character or word based, is dictated by factors like availability of a lexicon, grammar and the input source. In case of mathematical content, for instance, one may choose to use character recognition followed by structure analysis or else do both these steps together by making use of mathematical grammar constraints.

Handwriting recognition methods have found application in a varied set of domains. For instance, handwriting recognition methods have been used for music notes recognition [32], handwritten postal address and zip code recognition [24], forensics [106], digitization of forms [88], check processing [46] and facilitating easier input for devices like tablet PC [80], handwritten mathematics calculator [59] and a pen-based electronic version of games like Sudoku and crosswords. Each such domain poses a different set of challenges that must be addressed to fine-tune the recognition methods. In this dissertation, we make use of advances in the field of handwritten character recognition for handwritten mathematical content recognition from classroom videos. Specifically, our system can readily incorporate any character recognizer that is capable of recognizing handwritten characters and can produce a sorted list of top recognition results for each input character.

8.1.2 Text Extraction and Recognition from Videos

Indexing and retrieval of lecture recordings, e-learning initiatives [26], video-taped presentations [52] and commercial and personal videos are a few of the application domains that are making it ever important to recognize the textual content that is contained in multimedia recordings, especially the videos. Towards this end, a

number of methods that focus on detecting both handwritten and printed text in the videos have been proposed. A method for detecting and tracking text in digital video is proposed in [63], which implements a scale-space feature extractor that feeds an artificial neural processor to detect text blocks. More recently, a method for detecting and extracting text from broadcast videos was proposed by Peng *et al.* [78]. The method makes use of corners and geometrical features, determined using edge extraction, for support vector machine based initial classification and a subsequent examination of spatial inter-dependencies of different regions in the image is then used to detect and extract the text. In its current form, the solution described in this dissertation makes certain assumptions about the nature of the classroom video to facilitate easy extraction of text from the video. However, the use of advanced text extraction techniques, like the ones discussed above can help our solution to handle videos that do not necessarily conform to the simplifying assumptions.

Researchers have also proposed methods to partially process the video content (*e.g.* detect key frames) for easy retrieval, to acquire data about the movement of the tip of the pen (or pen strokes) on the whiteboard and to augment the whiteboard environment with cameras for tasks like controlling the computer [100, 92]. A system that automatically produces a set of key frames representing all the written content on the whiteboard before each erasure is described in [36]. The system could be used to manage the captured content, meetings in this case, efficiently. A system that classifies the video frame pixels into whiteboard background, pen strokes and foreground objects is proposed in [37]. The system uses this information to extract newly written pen strokes and here, the primary motivation for extracting the pen strokes is that they can then be used to efficiently transmit the whiteboard contents to a remote location. While a video camera based method to capture annotations made on PowerPoint slides projected on a whiteboard, is presented in [64, 65], a similar method that recognizes annotations made over printed documents is proposed in [94].

A fully-online, vision based interface for capturing handwriting and sketching by following the trajectory of the tip of the pen is presented in [73]. Bunke *et al.* [10] and also the Video Tablet project [71] describe methods that could be used to reconstruct the movement of the tip of the pen from a sequence of acquired images.

The issue of segmentation and recognition of text embedded in video sequences has been addressed by several researchers [40, 16, 66, 95, 79, 54, 50, 124]. While Chen *et al.*. [16] propose a probabilistic algorithm for video text extraction that is based on Bayesian adaptive thresholding and Monte-Carlo sampling, a gradient difference based method for text detection in videos is presented in [95]. A Laplacian method for video text detection is discussed in [79]. A text processing method for effective extraction of text from e-learning videos is proposed in [101]. Here, the method makes use of change frame detection and proposes a new text extraction algorithm. Wienecke at al. [123] have proposed and designed a system for recognizing unconstrained handwritten text from a video. Their main contribution is in the development of an incremental processing strategy that facilitates the recognition of portions of text as soon as they appear on the whiteboard. An interesting technique capable of spotting the most likely discussed chapters and topic words for each frame in a classroom video is described in [105, 103]. More recently, Öksüz *et al.*. [77] have proposed a video based text and equation editor for LaTeX. Their system makes use of a USB camera to capture and record the movement of the pen tip on a regular paper, and uses this information to generate LaTeX code for the text and equations written on the paper.

8.1.3 Mathematical Content Recognition

Mathematical content recognition [14] presents challenges that, in some aspects, are quite distinct from the challenges faced during the recognition of textual content. This is due to the fact that mathematical characters can be of different sizes and also

due to the fact that the content is spatially arranged as a complex two-dimensional structure. In general, the problem of mathematical content recognition can be divided into three main stages: character segmentation, character recognition and structure analysis. Character segmentation is an important preprocessing step before character recognition can occur and researchers have developed specialized (using structure information, for instance) character segmentation schemes for mathematical content [107]. For mathematical character recognition [69], researchers have addressed issues that arise from factors like the large number of similar symbols that must be distinctly recognized and the lack of a lexicon that could be used for final validation. The main challenge in structure analysis is the task of discriminating between the baseline, the subscript and the superscript characters. Zanibbi *et al.* [126] search for linear structures in the mathematical content to establish the baseline, which is then used by their approach to find secondary linear structures. Aly *et al.* [2], in their recent work evaluate the usefulness of geometrical information for the task of discriminating baseline, subscript and superscript. Earlier, Anderson [3] introduced a syntactic recognition method that is based on a coordinate grammar. A two-dimensional stochastic context-free grammar for syntactic recognition that takes pixel information from a scanned image is presented in [17].

Researchers have also shown that recognition accuracy for the mathematical content can be improved by combining two or more stages that are involved in mathematical content recognition. Prusa *et al.* [83] have proposed a method that combines the character segmentation and structural analysis stages to achieve better recognition accuracy for mathematical equations. Similarly, a hidden Markov model based method that avoids segmentation during pre-processing by making use of simultaneous segmentation and recognition capabilities is presented in [57, 55]. Awal *et al.* [5, 6] describe an interesting approach for simultaneous optimization of segmentation, recognition and structure analysis under the restriction of a mathematical

expression grammar.

It is a well established fact that QWERTY keyboards do not provide a very convenient interface for entering mathematical content. As a result, a number of researchers have focused on the problem of recognizing, both online and offline, handwritten mathematical content. Toyozumi *et al.* [107] present an expanded system for the recognition of handwritten mathematical equations that are provided to their system as strokes drawn on a tablet. Their system recognizes components of mathematical equations based on the position and combination of drawn strokes. Zanibbi *et al.* [126] have proposed a tree transformation based method for inferring the layout of mathematical content, both handwritten and typeset. A method for efficiently recognizing handwritten mathematical symbols from large data sets is presented in [121]. The method achieves efficiency by making use of a pre-classification strategy and 'elastic matching' to prune some symbols based on character features. More recently, Li *et al.* [62] have proposed an online method for recognition of handwritten mathematical content. Their approach is based on a spacing algorithm that uses recognized characters, character sizes and character locations and is also capable of recognizing matrices. A rule-based approach for improving the accuracy of handwritten mathematical content is presented in [53]. The approach makes use of certain types of contextual information associated with the characters. Correction facilities for misrecognized symbols are described in [56, 27].

Among other relevant ongoing efforts for mathematical content recognition, the Infty Project [47] is a voluntary research and development organization consisting of researchers from different universities and research institutes with focus on recognizing scientific content. Notable contributions from the project include InftyReader, the integrated mathematical document reader based on OCR and the InftyEditor, which is an authoring tool for mathematical documents. The Web Equation editor from Vision Objects [122] provides an online interface for writing mathematical equations

and visualizing the output in the form of LaTeX. The Detexify[2] project [20] provides an easy to use interface for searching LaTeX representation of handwritten mathematical symbols. To conduct the search the user simply needs to draw the symbol to be searched on the provided online canvas.

8.2 Speech Recognition

Speech recognition is the process of converting an acoustic signal, obtained using a microphone or a similar device, to a set of words. Over the last few decades speech processing technology has made very significant advances [60]. As a result, there are a number of commercial implementations of the technology including Dragon Naturally Speaking [21], Microsoft Speech [72] and IBM ViaVoice [118]. More recently, the commercial success of iPhone Siri [96], a voice enabled intelligent personal assistant for Apple's iOS, has brought the benefits of multiple years of speech recognition research to the general public. In the academic community, there have been some very active research contributions in the form of the HTK [39] project from the University of Cambridge and the Sphinx [98] project from CMU.

Modern day automatic speech recognition systems have come to largely rely on hidden Markov models (HMM) [85] based approaches. The adoption of HMMs has spurred significant performance improvements [41, 42] in the field of speech recognition. In the past, researchers have also used artificial neural networks (ANN) [108], finite state networks (FSN) [11] and support vector machines (SVM) [30] for facilitating speech recognition. In the work presented in this dissertation, our attempt is to use the capabilities of an existing speech recognition system to resolve some of the ambiguities in the output of a video based text recognizer. Given the fact that the audio content, which accompanies the handwritten mathematical content from classroom videos, does not necessarily conform to a strict mathematical speech grammar and is often interspersed with explanation, the use of recognizers that make use of language

models poses a number of challenges and restricts their wide applicability. In this dissertation, we present two approaches for integrating the audio recognition results into the final recognition results: (1) we make use of a phonetic search tool [74], which works well for non-standard grammar patterns without any training and is used to corroborate or contradict the presence of a certain character recognition result, and (2) from a classroom video, we use the segment of audio corresponding to a single handwritten equation, assume that it follows a known base mathematical speech grammar and use a speech recognizer [98] configured with a constrained grammar (based on character recognition results) to assist in the recognition process.

In the past, researchers have attempted to recognize mathematical content by means of speech recognition only or enable mathematical input by means of speech recognition. Mathifier [8], for instance, is an attempt to enable speech-only recognition for mathematical equations. Mathifier, which is based on Sphinx-4 constricts the speech recognition to math equations and attempts to recognize components of a equation with multiple smaller grammars. It is not entirely clear from the publication, how the authors arrive at such smaller grammars, termed loose grammars, and subsequently the lazy grammar. Our approach, by virtue of having the character recognition results available, is able to use the character recognition results to constrain the grammar for speech recognition. Another similar work [22], also allows for grammatically incomplete spoken phrases to enable entry of mathematical content. In the research presented in this dissertation, we assume presence of classroom videos and during the video preprocessing we have the ability to segment complete equations and extract complete speech corresponding to a handwritten equation. This enables us to handle grammar-assisted recognition of spoken content. In scenarios, where the instructor does not necessarily follow a known mathematical grammar, our approach that relies on phonetic search can be used.

8.3 Classifier Combination

The field of classifier combination has existed for quite some time now, and has been constantly evolving to address the challenges posed by new application domains. A fairly comprehensive survey of classifier combination techniques is presented in [109], which partitions the classifier combination methods along several distinct dimensions, some of which include - the way in which the output of the classifiers is combined, whether the number of classifiers is fixed or if the combination methods use classifiers from a large pool of classifiers, etc. There is also a large body of existing research focusing on generic methods for classifier combination. Lucey *et al.* [67], for instance, have proposed a theoretical framework for independent classifier combination . While a certain subset of classifier combination methods use another classifier to combine the output of multiple classifiers, another subset makes use of rules and functions to combine the output. Some of the combination techniques proposed in our research are adaptations of well known methods like weighted combination, Borda count [110] and other decision combination techniques [38].

Classifier combination techniques have also been applied to improving the recognition accuracy of handwriting recognizers [97, 29, 120, 86] and speech recognizers [25]. However, as compared to the classifiers combination approaches where all the classifiers accept the same input data and therefore produce output that could be easily combined, our classifiers not only accept different input data but also produce output that is not amenable to direct combination. It may be noted that similar problems with output not being amenable for direct combination may also occur when using advanced combination methods like bagging [9] and boosting [93]. For such methods, it is often hard to establish correspondence between the output of the various classifiers. A similar alignment issue for combining outputs of coreference resolution classifiers has been addressed by Vemulapalli *et al.* [116, 117], by generating a bipartite graph representation for the outputs of the classifiers and solving the maximum bipartite

matching problem. Classifier combination techniques more specifically, have also been applied to the recognition of handwritten mathematical content as well. Sabourin *et al.* [89] have used Bayesian combination, Dempster-Shafer evidential reasoning and dynamic classifier selection for combining the output of two handwritten digits recognizers. Garain *et al.* [31] present a multiple-classifier system for printed mathematical symbols that consists of hierarchically arranged classifiers. Cecotti *et al.* [12] focus on applying classifier combination methods for handwritten digit recognition to a set of classifiers that already have a fairly high recognition accuracy. It may be noted that the character recognizer (contained in the video text recognizer) in our system can be replaced by a character recognizer that is implemented as a multiple classifier system as long as it generates multiple ranked video options as output.

8.3.1 Audio-Video Based Recognition

Audio and video signals often carry complementary information and often errors in the audio recognition may not be accompanied by errors in the video recognition for the same segment(or character, in this case). Therefore, a combination of both information sources can potentially lead to a significant improvement in the recognition accuracy of both the audio and the video signal. Yu *et al.* [125, 49] propose a classifier combination framework for grammar-guided sentence recognition, and provide the results for the task of spoken email command recognition, where an acoustic classifier and a visual classifier (for recognizing lip movement, and tongue and teeth visibility) are combined. The Speech Pen system [58], for classrooms equipped with advanced digital whiteboards, recognizes speech and handwriting in the background and provides the instructor with predictions for further writing. A fairly comprehensive collection of research in the field of audio-visual speech recognition is presented in [82]. Anderson *et al.* [4], in the context of classroom videos that utilize slides and digital ink, present an empirical basis for addressing problems such as automatic

generation of full text transcripts for lectures. Their approach relies on matching spoken content with slide content, and recognizing the meaning of the content handwritten by the instructor using digital ink. An investigation of a number of strategies for combining HMM classifiers for improving audio-visual recognition is presented in [68], and a recommendation for using a hybrid of the sum and product rules is made based on empirical, theoretical and heuristic evidence. An interesting approach to using audio-visual inputs for celebrity recognition in unconstrained web-videos is presented in [91]. Humm *et al.* [43] present another interesting application that combines acquisition of online pen and speech signals for user authentication. It is clear that there is a significant amount of ongoing activity in the field of audio-video based recognition, which can be attributed to the complementary nature of the information contained in the audio and the video streams.

Hunsinger *et al.* [45] have proposed a multimodal mathematical formula editor which combines speech and handwriting recognition and describe the speech understanding module of the proposed system. In a later publication, Hunsinger *et al.* [44] present a multimodal probabilistic grammar that incorporates the syntactic-semantic attributes of spoken and handwritten mathematical formulas. A system for speech and handwriting data fusion based isolated mathematical symbol recognition is presented in [70]. Although neither the speech nor the handwritten data originates from a video, the approach for combination techniques used for combining the output of the character recognizer and the speech recognizer share commonality with the techniques presented in this dissertation. However, note that issues like ambiguity detection and A/V synchronization are not considered in the aforementioned research. Another relevant research effort, closely tied to [70], relates to the creation of a data set [84] with handwritten and spoken mathematical content. Unfortunately, the data set contains static image segments containing handwritten content and the corresponding audio content is stored as separate files. The unavailability of video (and therefore

the timestamp information) for the handwritten content and the fact that the spoken content is in French make the data set unsuitable for our experiments.

CHAPTER IX

CONCLUSIONS & FUTURE WORK

This dissertation explores the feasibility and advantages of using a combination of audio and video based recognizers for handwritten mathematical content recognition from classroom videos. Towards this end, we proposed an end-to-end framework, consisting of multiple stages, that allows for the use of audio and video based recognizers for the aforementioned recognition problem. The multiple stages of the end-to-end framework allowed us to examine the various nuances associated with combination of audio and video based recognizers. Specifically, in the context of audio-video based handwritten mathematical content recognition, we focused on (1) video preprocessing that included detection of key frames, character segmentation and timestamping of the segmented characters, (2) detection of segmented characters that may have already been correctly recognized by the video based character recognizer and forwarding only a limited subset of video options, for characters that are believed to have been incorrectly recognized, for subsequent combination, (3) prior to application of combination methods, methods for synchronizing the recognition results from the video and audio based recognizers, (4) methods for combining the synchronized recognition results from the audio and video based recognizers, and finally (5) methods based on mathematical speech grammar, where initial recognition results from video and audio based recognizers are used to arrive at a constrained speech grammar, which is then used for character and structure disambiguation. To validate the effectiveness of the proposed techniques, we created and made use of a database of videos recorded in a classroom-like environment. Experiments conducted using such videos were used to demonstrate the significant improvements in handwritten mathematical

content recognition accuracy that can be achieved using our techniques.

While significant, the contributions of this dissertation are only a modest attempt at addressing a subset of an otherwise diverse set of research issues in fields such as: educational multimedia, e-learning, combination of audio and video based classifiers for recognition of content from videos, mathematical content recognition, etc. Specifically, it would be interesting to explore the applicability of techniques proposed in this dissertation in other related fields like surveillance and monitoring where video feeds may need to be correlated to the audio feeds, gesture recognition where gestures may be accompanied by audible commands and classrooms equipped with advanced whiteboards (e.g. ones which can track the tip of the pen, etc.). Another interesting extension of the work proposed in this dissertation would be to explore scenarios with a larger set of recognizable symbols and explore its impact on the recognition accuracy. Similar extensions for the grammar assisted audio-video based mathematical recognition technique could be explored as well, where the impact of more general mathematical grammar on recognition accuracy could be explored.

REFERENCES

[1] Aho, A. V. and Ullman, J. D., *Principles of Compiler Design (Addison-Wesley series in computer science and information processing)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1977.

[2] Aly, W., Uchida, S., and Suzuki, M., "A large-scale analysis of mathematical expressions for an accurate understanding of their structure," in *Document Analysis Systems, 2008. DAS '08. The Eighth IAPR International Workshop on*, pp. 549 –556, sept. 2008.

[3] Anderson, R. H., "Two-dimensional mathematical notations," *Syntactic Pattern Recognition Applications (K.S. FU)*, pp. 147–177, 1977.

[4] Anderson, R., Hoyer, C., Prince, C., Su, J., Videon, F., and Wolfman, S., "Speech, ink, and slides: the interaction of content channels," in *Proceedings of the 12th annual ACM international conference on Multimedia*, MULTIMEDIA '04, (New York, NY, USA), pp. 796–803, ACM, 2004.

[5] Awal, A.-M., Mouchere, H., and Viard-Gaudin, C., "Towards handwritten mathematical expression recognition," in *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on*, pp. 1046 –1050, july 2009.

[6] Awal, A.-M., Mouchère, H., and Viard-Gaudin, C., "A hybrid classifier for handwritten mathematical expression recognition," in *Document Recognition and Retrieval*, pp. 1–10, 2010.

[7] Baker, J. B., Sexton, A. P., and Sorge, V., "A linear grammar approach to mathematical formula recognition from pdf," in *Proceedings of the 16th Symposium, 8th International Conference. Held as Part of CICM '09 on Intelligent Computer Mathematics*, Calculemus '09/MKM '09, (Berlin, Heidelberg), pp. 201–216, Springer-Verlag, 2009.

[8] Batlouni, S., Karaki, H., Zaraket, F., and Karameh, F., "Mathifier - speech recognition of math equations," in *Electronics, Circuits and Systems (ICECS), 2011 18th IEEE International Conference on*, pp. 301 –304, dec. 2011.

[9] Breiman, L. and Breiman, L., "Bagging predictors," in *Machine Learning*, pp. 123–140, 1996.

[10] Bunke, H., Von Siebenthal, T., Yamasaki, T., and Schenkel, M., "Online handwriting data acquisition using a video camera," in *Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on*, pp. 573 –576, sept. 1999.

[11] Casacuberta, F., "Some relations among stochastic finite state networks used in automatic speech recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 7, pp. 691–695, 1990.

[12] Cecotti, H., Vajda, S., and Belaïd, A., "High performance classifiers combination for handwritten digit recognition," in *Proceedings of the Third international conference on Advances in Pattern Recognition - Volume Part I*, ICAPR'05, (Berlin, Heidelberg), pp. 619–626, Springer-Verlag, 2005.

[13] Celik, M. and Yanikoglu, B., "Probabilistic mathematical formula recognition using a 2d context-free graph grammar," *Document Analysis and Recognition, International Conference on*, vol. 0, pp. 161–166, 2011.

[14] Chan, K.-F. and Yeung, D.-Y., "Mathematical expression recognition: a survey," *International Journal on Document Analysis and Recognition*, vol. 3, pp. 3–15, 2000. 10.1007/PL00013549.

[15] Chang, F., Chen, C.-J., and Lu, C.-J., "A linear-time component-labeling algorithm using contour tracing technique," *Comput. Vis. Image Underst.*, vol. 93, no. 2, pp. 206–220, 2004.

[16] Chen, D. and Odobez, J.-M., "Video text recognition using sequential monte carlo and error voting methods," *Pattern Recognition Letters*, vol. 26, no. 9, pp. 1386 – 1403, 2005.

[17] Chou, P. A., "Recognition of equations using a two-dimensional stochastic context-free grammar.," *SPIE Visual Communications and Image Processing IV*, pp. 852–863, 1989.

[18] "Classroom Video Data Set," *http://users.ece.gatech.edu/~smita/dataset/*, 2012 (accessed August 20, 2012).

[19] Davis, L. D. and Mitchell, M., *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

[20] "Detexify - LaTeX symbol classifier," *http://detexify.kirelabs.org/classify.html*, 2012 (accessed August 20, 2012).

[21] "Nuance - Dragon NaturallySpeaking Speech Recognition Software," *http://www.nuance.com/naturallyspeaking/*, 2012 (accessed August 20, 2012).

[22] Elliot, C. and Bilmes, J. A., "Computer based mathematics using continuous speech recognition," in *Striking a Chord: Vocal Interaction in Assistive Technologies, Games, and More: CHI workshop on non-verbal acoustic interaction*, (San Jose, CA, USA), 2007.

[23] Fateman, R. J., Tokuyasu, T., Berman, B. P., and Mitchell, N., "Optical character recognition and parsing of typeset mathematics," *Journal of Visual Communication and Image Representation*, vol. 7, pp. 2–15, 1996.

[24] Favata, J. T., "Offline general handwritten word recognition using an approximate beam matching algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 9, pp. 1009–1021, 2001.

[25] Fiscus, J., "A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER)," in *Automatic Speech Recognition and Understanding, 1997. Proceedings., 1997 IEEE Workshop on*, pp. 347 –354, dec 1997.

[26] Friedland, G., Hurst, W., and Knipping, L., "Educational multimedia," *Multimedia, IEEE*, vol. 15, pp. 54–56, july-sept. 2008.

[27] Fujiyoshi, A., Suzuki, M., and Uchida, S., "Syntactic detection and correction of misrecognitions in mathematical ocr," in *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on*, pp. 1360 – 1364, july 2009.

[28] Fujiyoshi, A., Suzuki, M., and Uchida, S., "Grammatical verification for mathematical formula recognition based on Context-Free tree grammar," *Mathematics in Computer Science*, Mar. 2010.

[29] Gader, P. D., Mohamed, M. A., and Keller, J. M., "Fusion of handwritten word classifiers," *Pattern Recogn. Lett.*, vol. 17, no. 6, pp. 577–584, 1996.

[30] Ganapathiraju, A., Hamaker, J., and Picone, J., "Hybrid svm/hmm architectures for speech recognition," in *in Speech Transcription Workshop*, pp. 504–507, 2000.

[31] Garain, U., Chaudhuri, B. B., and Ghosh, R. P., "A multiple-classifier system for recognition of printed mathematical symbols," in *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 1 - Volume 01*, ICPR '04, (Washington, DC, USA), pp. 380–383, IEEE Computer Society, 2004.

[32] George, S. E., "Online pen-based recognition of music notation with artificial neural networks," *Comput. Music J.*, vol. 27, no. 2, pp. 70–79, 2003.

[33] "GOCR," *http://jocr.sourceforge.net/*, 2012 (accessed August 20, 2012).

[34] Gonzalez, R. C., Woods, R. E., and Eddins, S. L., *Digital Image Processing Using MATLAB (Second Edition)*. Gatesmark Publishing, 2009.

[35] Ha, J., Haralick, R. M., and Phillips, I. T., "Understanding mathematical expressions from document images," in *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 2) - Volume 2*, ICDAR '95, (Washington, DC, USA), pp. 956–, IEEE Computer Society, 1995.

[36] He, L.-W., Liu, Z., and Zhang, Z., "Why take notes? use the whiteboard capture system," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, vol. 5, pp. V – 776–9 vol.5, april 2003.

[37] He, L.-W. and Zhang, Z., "Real-time whiteboard capture and processing using a video camera for remote collaboration," *IEEE Transactions on Multimedia*, vol. 9, no. 1, pp. 198–206, 2007.

[38] Ho, T. K., Hull, J. J., and Srihari, S. N., "Decision combination in multiple classifier systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 1, pp. 66–75, 1994.

[39] "HTK," *http://htk.eng.cam.ac.uk/*, 2012 (accessed August 20, 2012).

[40] Hua, X.-S., Yin, P., and Zhang, H.-J., "Efficient video text recognition using multiple frame integration," in *International Conference on Image Processing*, vol. 2, pp. II–397 – II–400 vol.2, 2002.

[41] Huang, X. D. and Jack, M. A., "Semi-continuous hidden markov models for speech signals," pp. 340–346, 1990.

[42] Huang, X., Alleva, F., Hayamizu, S., Hon, H.-W., Hwang, M.-Y., and Lee, K.-F., "Improved hidden markov modeling for speaker-independent continuous speech recognition," in *HLT '90: Proceedings of the workshop on Speech and Natural Language*, (Morristown, NJ, USA), pp. 327–331, Association for Computational Linguistics, 1990.

[43] Humm, A., Hennebert, J., and Ingold, R., "Combined handwriting and speech modalities for user authentication," *Trans. Sys. Man Cyber. Part A*, vol. 39, pp. 25–35, Jan. 2009.

[44] Hunsinger, J. and Lang, M., "A single-stage top-down probabilistic approach towards understanding spoken and handwritten mathematical formulas," in *INTERSPEECH*, pp. 386–389, 2000.

[45] Hunsinger, J. and Lang, M., "A speech understanding module for a multimodal mathematical formula editor," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on*, vol. 6, pp. 2413 –2416 vol.4, 2000.

[46] Impedovo, S. and Lucchese, M. G., "Improving a bank-check processing system with new hmm-based algorithms," in *AIC'05: Proceedings of the 5th WSEAS International Conference on Applied Informatics and Communications*, (Stevens Point, Wisconsin, USA), pp. 251–255, World Scientific and Engineering Academy and Society (WSEAS), 2005.

[47] "Infty Project," *http://www.inftyproject.org/en/index.html*, 2012 (accessed August 20, 2012).

[48] Jain, A., Nandakumar, K., and Ross, A., "Score normalization in multimodal biometric systems," *Pattern Recognition*, vol. 38, no. 12, 2005.

[49] Jiang, X., Yu, K., and Bunke, H., "Classifier combination for grammar-guided sentence recognition," in *MCS '00: Proceedings of the First International Workshop on Multiple Classifier Systems*, (London, UK), pp. 383–392, Springer-Verlag, 2000.

[50] Jianyong, S., Xiling, L., and Jun, Z., "An edge-based approach for video text extraction," in *Computer Technology and Development, 2009. ICCTD '09. International Conference on*, vol. 2, pp. 331 –335, nov. 2009.

[51] "Java Speech Grammar Format," *http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/*, 2012 (accessed August 20, 2012).

[52] Ju, S. X., Black, M. J., Minneman, S., and Kimber, D., "Summarization of video-taped presentations: Automatic analysis of motion and gesture," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 8, pp. 686–696, 1998.

[53] Kim, K., Rhee, T. H., Lee, J. S., and Kim, J. H., "Utilizing consistency context for handwritten mathematical expression recognition," in *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on*, pp. 1051 –1055, july 2009.

[54] Kim, W. and Kim, C., "A new approach for overlay text detection and extraction from complex video scene," *Image Processing, IEEE Transactions on*, vol. 18, pp. 401–411, feb. 2009.

[55] Kosmala, A. and Rigoll, G., "On-line handwritten formula recognition using statistical methods," in *ICPR 1998*, pp. Vol II: 1306–1308, 1998.

[56] Kosmala, A., Rigoll, G., and Brakensiek, A., "On-line handwritten formula recognition with integrated correction recognition and execution," in *ICPR 2000*, pp. Vol II: 590–593, 2000.

[57] Kosmala, A., Rigoll, G., Lavirotte, S., and Pottier, L., "On-line handwritten formula recognition using hidden markov models and context dependent graph grammars," in *Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on*, pp. 107 –110, sept. 1999.

[58] Kurihara, K., Goto, M., Ogata, J., and Igarashi, T., "Speech pen: predictive handwriting based on ambient multimodal recognition," in *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, (New York, NY, USA), pp. 851–860, ACM, 2006.

[59] LaViola, Jr., J. J., "Calligraphic interfaces: An initial evaluation of mathpad2: A tool for creating dynamic mathematical illustrations," *Comput. Graph.*, vol. 31, no. 4, pp. 540–553, 2007.

[60] Lee, C.-H., Soong, F. K., and Paliwal, K. K., eds., *Automatic Speech and Speaker Recognition: Advanced Topics*. Norwell, MA, USA: Kluwer Academic Publishers, 1999.

[61] Lee, H.-J. and Lee, M.-C., "Understanding mathematical expressions using procedure-oriented transformation," *Pattern Recognition*, vol. 27, no. 3, pp. 447–457, 1994.

[62] Li, C., Zeleznik, R., Miller, T., and LaViola, J., "Online recognition of handwritten mathematical expressions with support for matrices," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pp. 1 –4, dec. 2008.

[63] Li, H., Doermann, D., and Kia, O., "Automatic text detection and tracking in digital video," *Image Processing, IEEE Transactions on*, vol. 9, pp. 147–156, jan 2000.

[64] Li, W., Tang, H., and Zhu, Z., "Automated registration of high resolution images from slide presentation and whiteboard handwriting via a video camera," in *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW '04. Conference on*, pp. 168 – 168, june 2004.

[65] Li, W., Tang, H., and Zhu, Z., "Vision-based projection-handwriting integration in classroom," in *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW '06. Conference on*, pp. 9 – 9, june 2006.

[66] Lienhart, R., Effelsberg, W., Iv, P. I., and Mai, R., "Automatic text segmentation and text recognition for video indexing," *ACM/Springer Multimedia Systems*, vol. 8, pp. 69–81, 1998.

[67] Lucey, S., Chandran, V., and Sridharan, S., "A theoretical framework for independent classifier combination," in *ICPR 2002. Proceedings of the 16th International Conference on Pattern Recognition*, (Quebec, Canada), 2002.

[68] Lucey, S., Chen, T., Sridharan, S., and Ch, V., "Integration strategies for audio-visual speech processing: Applied to text-dependent speaker recognition," *IEEE Trans. Multimedia*, vol. 7, pp. 495–506, 2005.

[69] Malon, C., Uchida, S., and Suzuki, M., "Mathematical symbol recognition with support vector machines," *Pattern Recognition Letters*, vol. 29, no. 9, pp. 1326 – 1332, 2008.

[70] Medjkoune, S., Mouchère, H., Petitrenaud, S., and Viard-Gaudin, C., "Handwritten and audio information fusion for mathematical symbol recognition," in *ICDAR*, pp. 379–383, 2011.

[71] Moriya, M., Hayashi, T., Tominaga, H., and Yamasaki, T., "Video tablet based on stereo camera - human-friendly handwritten capturing system for educational use," in *Advanced Learning Technologies, 2005. ICALT 2005. Fifth IEEE International Conference on*, pp. 909 – 911, july 2005.

[72] "Speech Server - Microsoft Corporation," *http://www.microsoft.com/SPEECH/*, 2012 (accessed August 20, 2012).

[73] Munich, M. and Perona, P., "Visual input for pen-based computers," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, pp. 313–328, mar 2002.

[74] "Nexidia," *http://www.nexidia.com/*, 2012 (accessed August 20, 2012).

[75] Okamoto, M. and Miao, B., "Recognition of mathematical expressions by using the layout structure of symbols," *Document Analysis and Recognition, International Conference on*, 1991.

[76] "OpenCV," *http://opencv.willowgarage.com/wiki/*, 2012 (accessed August 20, 2012).

[77] Özcan Öksüz, Güdükbay, U., and Çetin, A. E., "A video-based text and equation editor for latex," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 6, pp. 952 – 960, 2008.

[78] Peng, X., Cao, H., Prasad, R., and Natarajan, P., "Text extraction from video using conditional random fields," in *Proceedings of the 2011 International Conference on Document Analysis and Recognition*, ICDAR '11, (Washington, DC, USA), pp. 1029–1033, IEEE Computer Society, 2011.

[79] Phan, T. Q., Shivakumara, P., and Tan, C. L., "A laplacian method for video text detection," in *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on*, pp. 66 –70, july 2009.

[80] Pittman, J. A., "Handwriting recognition: Tablet pc text input," *Computer*, vol. 40, no. 9, pp. 49–54, 2007.

[81] Plamondon, R. and Srihari, S. N., "On-line and off-line handwriting recognition: A comprehensive survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 63–84, 2000.

[82] Potamianos, G., Neti, C., Luettin, J., and Matthews, I., "Audio-visual automatic speech recognition: An overview," in *Issues in Visual and Audio-Visual Speech Processing, G. Bailly, E. Vatikiotis-Bateson, and P. Perrier (Eds.)*, MIT Press, 2004.

[83] Prusa, D. and Hlavac, V., "Mathematical formulae recognition using 2d grammars," in *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, vol. 2, pp. 849 –853, sept. 2007.

[84] Quiniou, S., Mouchère, H., Saldarriaga, S. P., Viard-Gaudin, C., Morin, E., Petitrenaud, S., and Medjkoune, S., "Hamex - a handwritten and audio dataset of mathematical expressions," in *ICDAR*, pp. 452–456, 2011.

[85] Rabiner, L. R., "A tutorial on hidden markov models and selected applications in speech recognition," in *Proceedings of the IEEE*, pp. 257–286, 1989.

[86] Rahman, A. F. R., Alam, H., and Fairhurst, M. C., "Multiple classifier combination for character recognition: Revisiting the majority voting system and its variations," in *DAS '02: Proceedings of the 5th International Workshop on Document Analysis Systems V*, (London, UK), pp. 167–178, Springer-Verlag, 2002.

[87] Raja, A., Rayner, M., Sexton, A. P., and Sorge, V., "Towards a parser for mathematical formula recognition," in *MKM*, pp. 139–151, 2006.

[88] Rossetto, S., Varej ao, F., and Rauber, T. W., "An expert system application for improving results in a handwritten form recognition system," in *IEA/AIE '02: Proceedings of the 15th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, (London, UK), pp. 383–392, Springer-Verlag, 2002.

[89] Sabourin, M., Mitiche, A., Thomas, D., and Nagy, G., "Classifier combination for hand-printed digit recognition," in *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on*, pp. 163 –166, oct 1993.

[90] "Sanyo VPC-HD1A 720p High-Definition Digital Media Camera," *http://us.sanyo.com/Cameras-Camcorders-Previous-Models/VPC-HD1A-720p-High-Definition-Digital-Media-Camera*, 2012 (accessed August 20, 2012).

[91] Sargin, M., Aradhye, H., Moreno, P., and Zhao, M., "Audiovisual celebrity recognition in unconstrained web videos," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pp. 1977 –1980, april 2009.

[92] Saund, E., "Bringing the marks on a whiteboard to electronic life," in *Proc. Cooperative BuildingsIntegrating Information, Organizations, and Architecture: Second Intl Workshop, CoBuild 99 (Lecture Notes in Computer Science*, pp. 69–78, Springer, 1999.

[93] Schapire, R. E., "The strength of weak learnability," *Mach. Learn.*, vol. 5, no. 2, pp. 197–227, 1990.

[94] Seok, J. H., Levasseur, S., Kim, K. E., and Kim, J. H., "Tracing handwriting on paper document under video camera," in *The 11th International Conference on Frontiers in Handwritting Recognition (ICFHR 2008)*, (Montreal, Canada), 2008.

[95] Shivakumara, P., Phan, T. Q., and Tan, C. L., "A gradient difference based technique for video text detection," in *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on*, pp. 156 –160, july 2009.

[96] "Apple - iPhone 4S - Ask Siri to help you get things done," *http://www.apple.com/iphone/features/siri.html*, 2012 (accessed August 20, 2012).

[97] Sirlantzis, K., Hoque, S., and Fairhurst, M. C., "Trainable multiple classifier schemes for handwritten character recognition," in *MCS '02: Proceedings of the Third International Workshop on Multiple Classifier Systems*, (London, UK), pp. 169–178, Springer-Verlag, 2002.

[98] "CMU Sphinx," *http://cmusphinx.sourceforge.net/*, 2012 (accessed August 20, 2012).

[99] "CMU Sphinx - Whitepaper," *http://cmusphinx.sourceforge.net/sphinx4/doc/Sphinx4Whitepaper.pdf*, 2012 (accessed August 20, 2012).

[100] Stafford-fraser, Q. and Robinson, P., "Brightboard: A video-augmented environment," in *Proceedings of the SIGCHI conference on Human factors in computing systems: common ground*, pp. 134–141, 1996.

[101] Sun, J., Katsuyama, Y., and Naoi, S., "Text processing method for e-learning videos," in *Computer Vision and Pattern Recognition Workshop, 2003. CVPRW '03. Conference on*, vol. 3, pp. 24 –24, june 2003.

[102] Sundaresan, C. S. and Keerthi, S. S., "A study of representations for pen based handwriting recognition of tamil characters," in *ICDAR '99: Proceedings of the Fifth International Conference on Document Analysis and Recognition*, (Washington, DC, USA), p. 422, IEEE Computer Society, 1999.

[103] Tang, L. and Kender, J., "Semantic indexing for instructional video via combination of handwriting recognition and information retrieval," in *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pp. 920 –923, july 2005.

[104] Tang, L., *Semantic content analysis and user interfaces for instructional video indexing*. PhD thesis, New York, NY, USA, 2006. Adviser-Kender, John R.

[105] Tang, L. and Kender, J., "Educational video understanding: mapping handwritten text to textbook chapters," in *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pp. 919 – 923 Vol. 2, aug.-1 sept. 2005.

[106] Tang, Y., Srihari, S. N., and Srinivasan, H., "Handwriting individualization using distance and rarity," in *Document Recognition and Retrieval*, 2012.

[107] Toyozumi, K., Yamada, N., Kitasaka, T., Mori, K., Suenaga, Y., Mase, K., and Takahashi, T., "A study of symbol segmentation method for handwritten mathematical formula recognition using mathematical structure information," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 2, pp. 630 – 633 Vol.2, aug. 2004.

[108] Tubach, J.-P. and Yoon, H., "Application of fully recurrent neural networks for speech recognition," in *ICASSP '91: Proceedings of the Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference*, (Washington, DC, USA), pp. 77–80, IEEE Computer Society, 1991.

[109] Tulyakov, S., Jaeger, S., Govindaraju, V., and Doermann, D., "Review of classifier combination methods," in *Studies in Computational Intelligence: Machine Learning in Document Analysis and Recognition*, pp. 361–386, 2008.

[110] van Erp, M., Vuurpijl, L., and Schomaker, L., "An overview and comparison of voting methods for pattern recognition," in *IWFHR '02: Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR'02)*, (Washington, DC, USA), p. 195, IEEE Computer Society, 2002.

[111] Vemulapalli, S. and Hayes, M., "Ambiguity detection methods for improving handwritten mathematical character recognition accuracy in classroom videos," in *Digital Signal Processing (DSP), 2011 17th International Conference on*, pp. 1 –6, july 2011.

[112] Vemulapalli, S. and Hayes, M., "Synchronization and combination techniques for audio-video based handwritten mathematical content recognition in classroom videos," in *Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on*, pp. 941 –946, nov. 2011.

[113] Vemulapalli, S. and Hayes, M., "Towards audio-video based handwritten mathematical content recognition in classroom videos," in *Communications, Computers and Signal Processing (PacRim), 2011 IEEE Pacific Rim Conference on*, pp. 774 –779, aug. 2011.

[114] Vemulapalli, S. and Hayes, M., "Audio-Video Based Handwritten Mathematical Content Recognition in Classroom Videos," *Integrated Computer-Aided Engineering*, (Under Review), Submitted April 2012.

[115] Vemulapalli, S. and H. Hayes III, M., "Using audio based disambiguation for improving handwritten mathematical content recognition in classroom videos," in *DAS 2010: The Tenth IAPR International Workshop on Document Analysis Systems*, 2010.

[116] Vemulapalli, S., Luo, X., Pitrelli, J. F., and Zitouni, I., "Classifier combination techniques applied to coreference resolution," in *NAACL '09: Proceedings of Human Language Technologies: The 2009 Annual Conference of*

the North American Chapter of the Association for Computational Linguistics, Companion Volume: Student Research Workshop and Doctoral Consortium, (Morristown, NJ, USA), pp. 1–6, Association for Computational Linguistics, 2009.

[117] Vemulapalli, S., Luo, X., Pitrelli, J. F., and Zitouni, I., "Using bagging and boosting techniques for improving coreference resolution," *Informatica (Slovenia)*, vol. 34, no. 1, pp. 111–118, 2010.

[118] "Embedded Viavoice," *http://en.wikipedia.org/wiki/IBM_ViaVoice*, 2012 (accessed August 20, 2012).

[119] Wang, J., "Pen computing: Digital ink and printed document," in *ICDAR '05: Proceedings of the Eighth International Conference on Document Analysis and Recognition*, (Washington, DC, USA), p. 334, IEEE Computer Society, 2005.

[120] Wang, W., Brakensiek, A., and Rigoll, G., "Combination of multiple classifiers for handwritten word recognition," in *IWFHR '02: Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR'02)*, (Washington, DC, USA), p. 117, IEEE Computer Society, 2002.

[121] Watt, S. M. and Xie, X., "Recognition for large sets of handwritten mathematical symbols," in *ICDAR*, pp. 740–744, 2005.

[122] "Web Equations," *http://webdemo.visionobjects.com/equation.html*, 2012 (accessed August 20, 2012).

[123] Wienecke, M., Fink, G. A., and Sagerer, G., "Toward automatic video-based whiteboard reading," *IJDAR*, vol. 7, no. 2–3, pp. 188–200, 2005.

[124] Yi, J., Peng, Y., and Xiao, J., "Using multiple frame integration for the text recognition of video," in *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on*, pp. 71 –75, july 2009.

[125] Yu, K., Jiang, X., and Bunke, H., "Combining acoustic and visual classifiers for the recognition of spoken sentences," in *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, vol. 2, pp. 491 –494 vol.2, 2000.

[126] Zanibbi, R., Blostein, D., and Cordy, J. R., "Recognizing mathematical expressions using tree transformation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 11, pp. 1455–1467, 2002.

VITA

Smita Vemulapalli is a Ph.D. candidate at the School of Electrical and Computer Engineering, Georgia Institute of Technology. She is also a member of the Center for Signal and Image Processing (CSIP) which conducts research in speech and image processing, digital signal processing algorithms, digital signal processing software and hardware architectures. As a Ph.D. student, Smita spent a summer working as a research intern with the natural language processing group at IBM Thomas J. Watson Research Center and another summer as an intern with the educational and productivity solutions group at Texas Instruments. Her research interests include pattern recognition, computer vision and digital signal processing applications. She holds an M.S. in Electrical and Computer Engineering from Georgia Institute of Technology and a B.E. in Electronics and Communication Engineering from PSG College of Technology, Coimbatore, India.